A GMM primer with focus on speech processing Version 1*

Thomas Hain
University of Sheffield, UK

May 2020

1 Introduction

Gaussian mixture models (GMMs are a simple, yet powerful generative model that can be used for speech processing, to describe the production of speech on frame levels. GMMs are timeless model, i.e. they assume that all samples drawn are identically distributed (i.i.d). Traditionally speech is considered to be a sequence of data points which are highly interlinked. Therefore GMMs are used only in certain circumstances.

There are many great references for GMMs, Chris Bishop's book [1] among the first to mention. These references do well in explaining the mathematical properties and use for machine learning. Here we follow a slightly different description.

Generative models aim to provide path to generating data that has distributions similar to the one found in the real samples. Given the model parameters λ the observations are drawn from a distribution composed of a weighted sum of Gaussians.

$$p(x|\lambda) = \sum_i w_i \mathcal{N}(x; \mu, \Sigma)$$

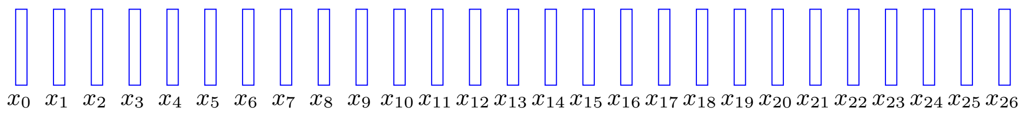
The number of Gaussians is flexible, but at least in standard settings fixed and assumed known. x of course is assume to be a vector , and therefore the normal distributions are assumed to be multi-variate distributions, with means being vectors and covariance matrices. In practice often diagonal covariances are assumed.

2 Modeling of speech with GMMs

The waveform is normally converted into a sequence of features, or to be precise, feature vectors.

$$\boldsymbol{X} = \boldsymbol{x}_1^T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T\}$$

In illustration the waveform turns into


x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{17} x_{18} x_{19} x_{20} x_{21} x_{22} x_{23} x_{24} x_{25} x_{26}

where each rectangle symbolizes a d dimensional feature vector. One could of course think of computing the likelihood of the whole sequence, however sequence lengths varies normally considerably. Therefore we make use of the frame independence assumption:

$$P(\boldsymbol{X}) = P(\boldsymbol{x}_1^T) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_1^{t-1}) \approx \prod_{t=1}^{T} p(\boldsymbol{x}_t) \tag{1}$$

The approximation is necessary because frame independence in speech is not reality. In most cases the feature vectors are some form of spectral slice representation (a slice in a spectrogram), for example consisting of MFCC features. One can observe that changes in spectrograms are smooth, clearly indicating dependence.

*This document is only for use in teaching. **ONLY FOR PERSONAL USE - NO REDISTRIBUTION TO ANY PERSON.** The document at this stage is incomplete and misses amongst other things a considerable number of references, while other references are incomplete. This will be remedied in future versions.

2.1 GMMs

The likelihood of a frame assumed to be derived from a Normal (Gaussian) distribution, or better a family of distributions. The random process of generating a sample starts by choosing a a Gaussian from a fixed set of M Gaussians. That selection process is governed by probabilities:

$$P(g_t = j) = P(g = j) = w_j$$

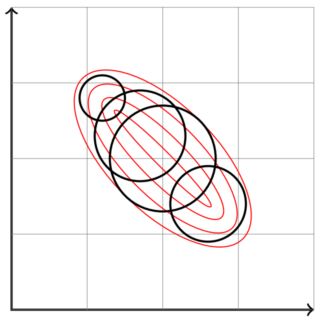
denotes the probability that the j-th Gaussian is chosen as Gaussian at time t,namely g_t. This of course is time-independent and we a assign a prior value or weight w_j. Now a sample is chosen from that Gaussian, according to its parametrisation, namely

$$p(\boldsymbol{x}_t|g_t = j) = \mathcal{N}\left(\boldsymbol{x}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \equiv \mathcal{N}\left(\boldsymbol{x}_t; \Theta_j\right)$$

The mean of course is vector valued, and the covariance matrix has to be a real symmetric positive definite matrix[1]. However, in practice this implies the estimation of a lot of parameters and therefore a requirement of a large amount of training data. Then assumptions such as diagonality for the covariance are used, which imply that all dimensions are independent, again an assumption hard to justify for speech.

$$p(\boldsymbol{x}_t|g_t = j) = \prod_{k=1}^{d} \mathcal{N}\left(x_{k,t}; \mu_{kj}; \sigma_{kj}^2\right)$$

However, this issue is somewhat addressed by the mixture model, which can be illustrated in two dimensional graph. As can be found in text books, the contour line of a 2D normal distribution is an ellipse. The angle of the ellipse is determined by the off diagonal elements of the covariance matrix. Naturally a diagonal covariance matrix has no off-diagonal elements, therefore the axes of the ellipse are aligned with the coordinates.



We can think of approximating a full covariance with a sum of diagonal covariance distributions. Note that the illustrations even makes use of diagonal covariances with identical elements in the covariance matrix, i.e. $\Sigma = \sigma^2 \boldsymbol{I}$.

2.1.1 Model fit

When modeling speech the number of mixture components M is assumed to be known. However in practice, experimentation should be conducted to find the appropriate number. This is the number that yields the best performance for the task at hand, on an independent test set. How well a model fits the data can be simply computed by computing the total likelihood of the data. Lets assume that we have a total of K utterances, each with length T_k. Then the total likelihood is simply given by

$$\mathcal{L}\left\{D|\lambda\right\}=p(X^{(1)}, X^{(2)}, \ldots, X^{(K)}|\lambda) = \prod_{k=1}^{K} p(X^{(k)}|\lambda) = \prod_{k=1}^{K}\prod_{t=1}^{T_k} p(x_t^{(k)}|\lambda) \tag{2}$$

Dmay be the training set or the test set. Any training algorithm should strive to find a solution that maximizes the total training set likelihood (not in all cases - see below). Note that by increasing the number of mixture components you are guaranteed to increase training set likelihood (at least assuming appropriate training strategies). This leads to overfitting.

Difference between test set likelihood and training set likelihood will give an indication on domain mismatch and over fitting.

[1] A matrix $\boldsymbol{A}$is positive definite if the term $\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z}$is always positive for any non-zero z.

2.1.2 Computational considerations

Computing the likelihood of data requires the multiplication of a a lot of small numbers, thereby leading very quickly to numbers out of range of floating point precision in computers. For many cases a simple solution is to operate all computation in the log domain (natural log is most convenient in practice). The log-likelihood of the data is given by

$$\log \mathcal{L}\{D|\lambda\} = \sum_k \sum_{t=1}^{T_k} \log p(\boldsymbol{x}_t^{(k)}|\lambda)$$

For typical speech representations the typical log-likelihood value per frame (feature vector) ranges from -60 to -80. Summing these values is much easier to perform without numerical issues. Unfortunately the computation of the GMM contribution is not easy to simplify.

$$\log p(\boldsymbol{x}|\lambda) = \log \sum_{j=1}^{M} w_j p(\boldsymbol{x}|\lambda_j)$$

While taking the log of the normal distribution leads to simple computation, namely a quadratic form,

$$\log p(\boldsymbol{x}|\Theta_j) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$$

the summation over all components cannot be be performed in the log-domain. Numbers need to be converted between those domains, with some attention to numerical issues (see the so called logadd functions).

2.2 Using GMMs for classification

Classification is the task of assigning a label, chosen from a finite set, to data. In the case of speech such labeling can be performed on different data portions. One can assign a label to each frame (feature vector), or to fragments of speech, such as phonemes, words, or complete utterances.

A simple and standard approach is to train a GMM for each class. Let us assume that L labels are available. Then one can obtain a GMM for each label, namely $\lambda_l$with $l = 1 \dots L$. Given a fragment of speech $\boldsymbol{X}$we would like to assign the most probable label.

$$\max_{l=1\dots L} P(c = l|\boldsymbol{X}) = \max_{l=1\dots L} p(\boldsymbol{X}|c = l)P(c = l)$$

In the above we naively applied Bayes rule - which implies that we somehow have access to the true values for the above probabilities. If we are in a position to ignore the prior, the classification task (finding label $c$for $\boldsymbol{X}$) is simply written as

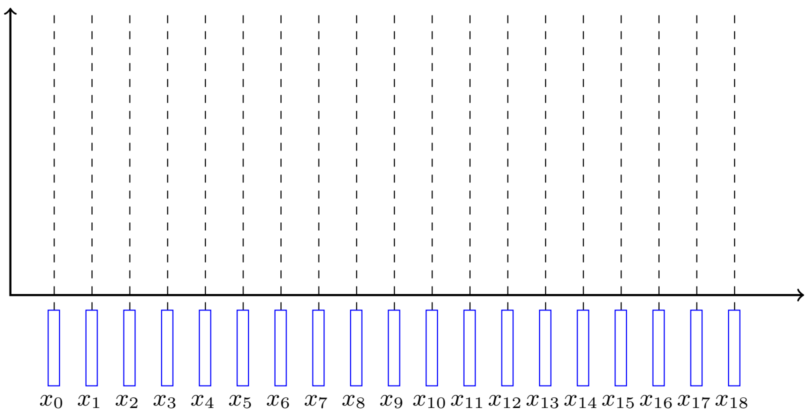
$$c = \arg \max_{l=1\dots L} \log p(\boldsymbol{X}|\lambda_l)$$

where we made use of the fact that finding the maximum is equivalent in the log domain, and that a class is represented by the model for that class, λ_l.

2.3 Using GMMs for unsupervised clustering

3 Frame level learning

3.1 Decision oriented learning - the Viterbi approach


x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{17} x_{18}

3.2 Soft decisions - E-M learning

3.3 An alternative way - Maximum a-posteriori (MAP)

3.4 Discriminative training

References

[1] Chris Bishop. Pattern Recognition and Machine Learning.

[2] Dempster and Laird (1977). The Expectaction Maximisation Algorithm.

[3] Gauvain and Lee (1994). Maxiumum a Posteriori Estimation.

[4] H. Hermansky (1990). MFCCs