Machine Learning

Lecture 2: GMM and EM

Lecturer: Haim Permuter *Scribe: Ron Shoham*

I. Introduction

This lecture comprises introduction to the Gaussian Mixture Model (GMM) and the Expectation-Maximization (EM) algorithm. Parts of this lecture are based on lecture notes of Stanford's CS229 machine learning course by Andrew NG[1]. This lecture assumes you are familiar with basic probability theory. The notation here is similar to that in Lecture 1.

II. Review: Supervised classification

In supervised classification, our target is to analyze the labelled training data we get, and to use it to generate a model to map and classify new examples. Below is a general model of supervised learning.

observation	**label**
r_1	c_1
r_2	c_2
$\vdots$	$\vdots$
r_N	c_N

Here r represents the raw observation vectors with their respective label c, and $\hat{c}$ is the estimation of the model.

1) Feature extraction: The goal of the feature extraction block is to extract the features that contribute most to the classification and to eliminate the rest. For instance, in speech recognition a well-known feature extraction technique is called Cepstrum, and in texture classification, it is based on the discrete cosine transform (DCT) or wavelet. All the above-mentioned features are based on the frequency domain. In some cases, one might also use dimension reduction in addition to these features, when, for example, the feature vector is too large or there is high redundancy. Two good feature reduction methods are PCA and Autoencoders.
2) Statistical model: The goal of the statistical model is to represent the statistics of each class, which allows the classes to be separated from each other. The statistical model usually has some probability justification (such as the GMM) but sometimes it might just be a separations technique of a different class. Usually a good statistical model can also be used for additional tasks such as data compression or denoising.
3) Decision: The decision component is responsible for using the statistical model output to classify the input. In some cases, we may generate more than one statistical model, and the decision component uses all of the outputs.

The GMM is a statistical model which assumes that every probability distribution can be approximated with a set of gaussians.

III. Mixtures of Gaussians and the EM algorithm

A. Gaussian Mixture Model (GMM) - Introduction

In supervised learning, GMM models the distribution of each class as a set of weighted gaussians, $P(x_i, z_i|c) = P(x_i|z_i, c)P(z_i|c)$, where z_i is a hidden random variable that is not observed. One assumption that is made is that any distribution can be well modelled by a set of gaussians. In the figure below, you can see three weighted gaussians in $\mathbb{R}^1$ and their sum which models the distribution of a random variable. Another assumption of the GMM is that any sample, $x \in \mathbb{R}^l$, was generated from a single gaussian.

$$P(x_i|z_i = j, c) = \frac{1}{\sqrt{(2\pi)^l|\Sigma_j|}} \exp(-\frac{1}{2}(x - \mu_j)^T\Sigma_j^{-1}(x - \mu_j)) \tag{1}$$

The probability of getting a specific gaussian given a class is

$$P(z_i = j|c) = \phi_j \tag{2}$$

Note that $\phi_j \geq 0$, $\forall j$ and that $\sum_{j=1}^{k} \phi_j = 1$.

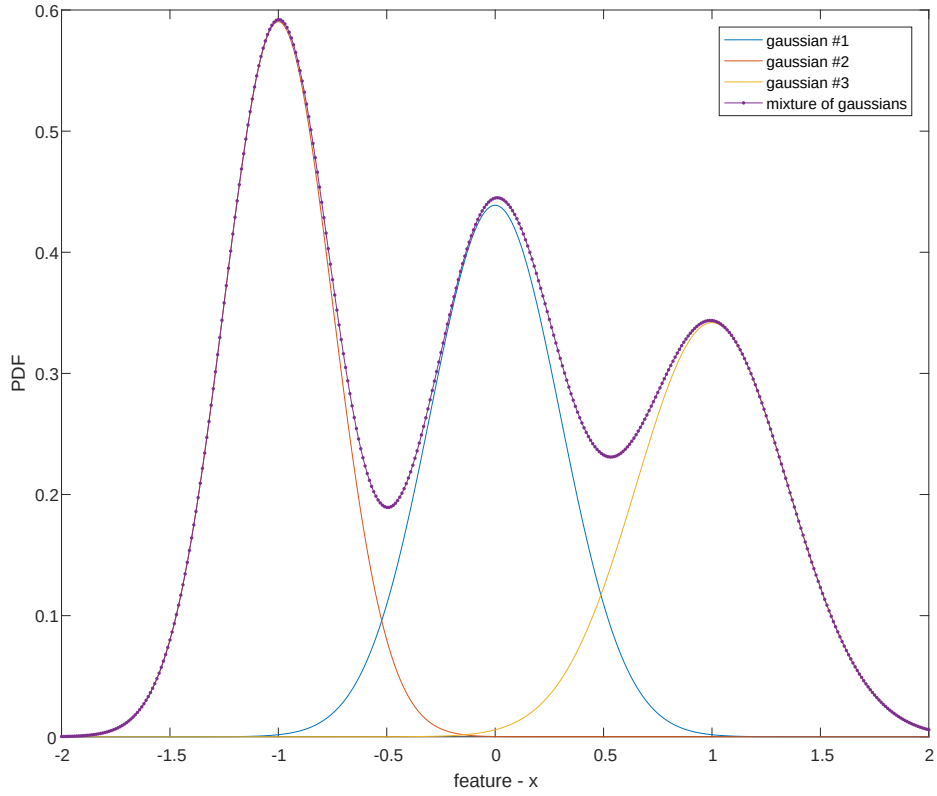

Fig. 1. Gaussian mixture model built from three weighted gaussians. The figure shows three weighted gaussians with different μ and σ. The sum represents the mixture model.

From now on we only refer to a specific class c, since each class has its own model. Our goal is to estimate the model's parameters ϕ_j, μ_j and Σ_j, $\forall j \in \{1, ..., k\}$, and we use *maximum likelihood*[2] criteria to maximize the expression $P(X^n; \theta)$, where θ represents the parameters.

$$\underset{\theta}{\operatorname{argmax}}\, P(x^n; \theta) = \underset{\theta}{\operatorname{argmax}} \log(P(x^n|\theta)) \tag{3a}$$

$$= \underset{\theta}{\operatorname{argmax}} \log(\prod_i P(x_i|\theta)) \tag{3b}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_i \log(P(x_i|\theta)) \tag{3c}$$

We define ℓ as the log-likelihood function. To estimate our parameters, we can write the log-likelihood of our data:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log P(x_i; \phi, \mu, \Sigma) \tag{4a}$$

$$= \sum_{i=1}^{m} \log \sum_{j=1}^{k} P(x_i|z=j; \mu_j, \Sigma_j) P(z=j; \phi_j, \mu_j, \Sigma_j) \tag{4b}$$

B. One Gaussian introduction and complete solution

First, let's try to model our data using a single Gaussian. This example is in one dimensional space.

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log \sum_{j=1}^{k} P(x_i|z=j; \mu_j, \sigma_j) P(z=j; \mu_j, \sigma_j) \tag{5a}$$

$$\overset{(a)}{=} \sum_{i=1}^{m} \log P(x_i|\mu, \sigma) \tag{5b}$$

$$= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{5c}$$

Where (a) follows from $k = 1$. Now lets take derivative in order to find the parameters which maximizes the log-likelihood. First we take derivative with respect to μ

$$\frac{\partial}{\partial \mu} \ell(\phi, \mu, \Sigma) = -\frac{1}{2\sigma^2} \sum_{i=1}^{m} (x_i - \mu) \tag{6a}$$

$$= 0 \tag{6b}$$

Therefore

$$\boxed{\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} x_i} \tag{7}$$

We perform the same thing with respect to σ^2

$$\frac{\partial}{\partial\sigma^2}\ell(\phi,\mu,\Sigma) = \frac{\partial}{\partial\sigma^2}\sum_{i=1}^{m}(-\frac{1}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}(x_i-\mu)^2) \tag{8a}$$

$$= -\frac{m}{2\sigma^2} + \sum_{i=1}^{m}\frac{(x_i-\mu)^2}{2(\sigma^2)^2} \tag{8b}$$

$$= 0 \tag{8c}$$

We get

$$\boxed{\hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu)^2} \tag{9}$$

C. GMM parameters estimation and EM motivation

Two Gaussians example: Now lets try to find an analytic solution to maximize the log-likelihood of two Gaussians model.

$$\ell(\phi,\mu,\Sigma) = \sum_{i=1}^{m}\log\sum_{j=1}^{k}P(x_i|z=j;\mu,\sigma)P(z=j;\theta) \tag{10a}$$

$$= \sum_{i=1}^{m}\log(\phi_1\mathcal{N}(x_i;\mu_1,\sigma_1^2) + \phi_2\mathcal{N}(x_i;\mu_2,\sigma_2^2)) \tag{10b}$$

ϕ_j stands for $P(z=j;\theta)$. By differentiating (10b) with respect to μ_1, we obtain

$$\sum_{i=1}^{m}\frac{1}{\phi_1\mathcal{N}(x_i;\mu_1,\sigma_1^2) + \phi_2\mathcal{N}(x_i;\mu_2,\sigma_2^2)}\phi_1\mathcal{N}(x_i;\mu_1,\sigma_1^2)\frac{x_i-\mu_1}{\sigma^2} = 0 \tag{11}$$

This we cannot solve in a closed form to get a clean maximum likelihood expression. Therefore, we use the Expectation-Maximization (EM) algorithm.

D. Formal look at the EM algorithm[7]

The EM algorithm was first explained in a 1977 paper, *Maximum Likelihood from Incomplete Data via the EM Algorithm* [3]. It is an iterative algorithm for using maximum likelihood to estimate the parameters of a statistical model with unobserved (hidden)

variables (also called *latent variables* where lateo is "lie hidden" in Latin). It has two main steps. First is the E-step, which stands for expectation. We compute some probability distribution of the latent variables so we can use it for expectations. Second comes the M-step, which stands for maximization. The EM algorithm find a local maximum and that depends on the initial model that we start with.

Problem Definition: Let x^n be a random vector distributed i.i.d. that we observe. Let Z^n be a hidden vector that X^n depends on. s.t. $P(x^n|z^n) = \prod_{i=1}^n p(x_i|z_i)$ also distributed i.i.d.. i.e., $P(z^n) = \prod_{i=1}^n P(z_i)$. The distribution of X^n and Z^n have some parameters θ that we are interested to find. In GMM, for instance, Z_i stands for the Gaussian number of sample i, i.e., $z_i \in \{1, 2, ..., k\}$ where k is the total number of Gaussians. Our goal is to maximize the log-likelihood

$$\log P_{X^n}(x^n; \theta) = \log \left(\sum_{z^n} P_{X^n, Z^n}(x^n, z^n) \right) \tag{12}$$

In a shorter notation simply write

$$\log(x^n; \theta) = \log \left(\sum_{z^n} P(x^n, z^n) \right). \tag{13}$$

EM Algorithm: The following algorithm is the EM algorithm that is an iterative algorithm and is based on two steps: Expectation and Maximization steps.

EM Algorithm:

---

1) **function** EM($x^n, \theta^{(0)}$)
2) **for** iteration $t \in 1, 2, ...$ **do**
3) $Q^{(t)}(z_i) = P(z_i|x_i; \theta^{(t-1)}) \quad \forall i = 1, 2, ..., n \quad \forall z_i \in \mathcal{Z}$ **E-step**
4) $\theta^{(t)} = \text{argmax}_\theta \, \mathbb{E}_{Q^{(t)}_{Z^n}}[\log(P(x^n, Z^n; \theta))]$ **M-step**
5) **if** $\log P(x^n|\theta^{(t)}) - \log P(x^n|\theta^{(t-1)}) < \epsilon$ **then**
6) return $\theta^{(t)}$

The stopping criteria can be changed. Can be either the one that is written, $\log P(x^n|\theta^{(t)}) - \log P(x^n|\theta^{(t-1)}) < \epsilon$, or a fixed number of iterations or $||\theta^{(t)}) - \theta^{(t-1)})|| < \epsilon$ for some norm $||\cdot||$.

Derivation of the EM algorithm: Now, we are going to show that in each iteration the algorithm increases the log likelihood, i.e., $\log P(x^n|\theta^{(t)})$ increases as t increases. Hence, the EM algorithm convergence to a local minimum that depends on the initial model $\theta^{(0)}$.

For any $Q(z^n)$ we have

$$\log P(x^n;\theta^{(t)}) = \log\left(\sum_{z^n} Q(z^n)\frac{P(x^n,z^n;\theta^{(t)})}{Q(z^n)}\right) \tag{14}$$

$$= \log\left(\mathbb{E}_{Q_{Z^n}}\left[\frac{P(x^n,Z^n;\theta^{(t)})}{Q(Z^n)}\right]\right) \tag{15}$$

$$\overset{\text{(a)}}{\geq} \mathbb{E}_{Q_{Z^n}}\left[\log\ \frac{P(x^n,Z^n;\theta^{(t)})}{Q(Z^n)}\right] \tag{16}$$

Where (a) follows Jensen's inequality. Now, using algebra and information measure we obtain

$$\mathbb{E}_{Q_{Z^n}}\left[\log\ \frac{P(x^n,Z^n;\theta^{(t)})}{Q(Z^n)}\right] = \mathbb{E}_{Q_Z}\left[\log(P(x^n;\theta^{(t)}))\right] + \mathbb{E}_{Q_{Z^n}}\left[\log\frac{P(Z^n|x^n;\theta^{(t)})}{Q(Z^n)}\right]$$

$$= \log(P(x^n;\theta^{(t)})) + \mathbb{E}_{Q_{Z^n}}\left[\log\frac{P(Z^n|x^n;\theta^{(t)})}{Q(Z^n)}\right] \tag{17}$$

$$= \log(P(x^n;\theta^{(t)})) - D(Q_{Z^n}||P_{Z^n|x^n,\theta^{(t)}}) \tag{18}$$

Hence, we can choose $Q(z^n) = P(z^n|x^n,\theta^{(t)})$, i.e., $Q(z_i) = P(z_i|x_i,\theta^{(t)})$ and we obtain that (a) in (16) is with equality. So now when $Q(z^n)$ is fixed, we can maximize over all θ the following expression

$$\theta^{(t+1)} = \arg\max_\theta \mathbb{E}_{Q_{Z^n}}\left[\log\frac{P(x^n,Z^n;\theta)}{Q(Z^n)}\right] = \arg\max_\theta \mathbb{E}_{Q_{Z^n}}\left[\log P(x^n,Z^n;\theta)\right],$$

Hence we have

$$\log(P(x^n;\theta^{(t+1)})) - D(Q_{Z^n}||P_{Z^n|x^n,\theta^{(t+1)}}) \overset{(a)}{=} \mathbb{E}_{Q_{Z^n}}\left[\log\frac{P(x^n,Z^n;\theta^{(t+1)}}{Q(Z^n)}\right]$$

$$\overset{(b)}{\geq} \mathbb{E}_{Q_{Z^n}}\left[\log\frac{P(x^n,Z^n;\theta^{(t)}}{Q(Z^n)}\right]$$

$$\stackrel{(c)}{=} \log P(x^n;\theta^{(t)})$$

where (a) and (c) follows from the E step derivation, (b) from the M step. Finaly the last sequence of equations implies that $\log P(x^n;\theta^{(t+1)}) \geq \log P(x^n;\theta^{(t)})$ because divergence is non negative.

E. Expectation Maximization (EM) application for GMM

Let's start with the E-step. We define weights $w(j,i) \stackrel{\triangle}{=} P(z=j|x_i;\theta)$. The weights are 'soft' assignment of the sample x_i to the Gaussian j.

$$w(j,i) = P(j|x_i;\theta) \tag{19a}$$
$$= \frac{P(j,x_i;\theta)}{P(x_i;\theta)} \tag{19b}$$
$$= \frac{P(j;\theta)P(x_i;j,\theta)}{\sum_l P(x_i,l;\theta)} \tag{19c}$$
$$= \frac{P(j;\theta)N(x_i;\mu_j,\sigma_j^2)}{\sum_l P(l;\theta)N(x_i;\mu_l,\sigma_l^2)} \tag{19d}$$

Now, we wish find the new parameters θ which maximize the log-likelihood with respect to the expectations (the M-step). First let's define $\phi(j) \stackrel{\triangle}{=} P(j;\theta)$. The model parameters are $\theta = \{\phi(j),\mu_j,\sigma_j^2\}$.

$$\underset{\theta}{\text{argmax}}\mathbb{E}_{Q(Z)}[\log(P_{X,Z}(x,z;\theta))] = \tag{20}$$
$$= \underset{\phi(j),\mu_j,\sigma_j^2}{\text{argmax}} \sum_{i=1}^{m}\sum_j w(j,i)\log P(j,x_i;\theta) \tag{21}$$
$$= \underset{\phi(j),\mu_j,\sigma_j^2}{\text{argmax}} \sum_{i=1}^{m}\sum_j w(j,i)\log(\phi(j)N(x_i;\mu_j,\sigma_j^2))$$
$$= \underset{\phi(j),\mu_j,\sigma_j^2}{\text{argmax}} \sum_{i=1}^{m}\sum_j w(j,i)(\log\phi(j) + \log\frac{1}{\sqrt{2\pi\sigma_j^2}} - \frac{(x_i-\mu_j)^2}{2\sigma_j^2})$$

Deriving with respect to μ_j and comparing to 0 gives

$$\mu_j = \frac{\sum_i w(j,i)x_i}{\sum_i w(j,i)} \tag{22}$$

Performing the same thing with respect to σ_j^2 gives

$$\sigma_j^2 = \frac{\sum_i (x_i - \mu_j)^2 w(j,i)}{\sum_i (w(j,i))} \tag{23}$$

Finding the optimal $\phi(j)$ is a bit more difficult since it is a probability distribution function and therefore it has some constrains. The constrains are

$$\sum_j \phi(j) = 1$$

$$\phi(j) \geq 0 \quad \forall j$$

The solution for this comes from the field of convex optimization. Further discussion about convex optimization problems and Lagrange multipliers can be found in the *appendix* and in Boyd's book[4]. Let's define the problem as a standard convex optimization problem (defintion 3)

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^{m}\sum_{j=1}^{k} w(j,i)\log\phi(j) \\ \text{subject to} \quad & -\phi(j) \leq 0, \quad 1 \leq j \leq k \\ & \sum_j \phi(j) = 1 \end{aligned}$$

The Lagrange multipliers (definition 4) are

$$L(\phi(j), \lambda, \nu) = -\sum_{i=1}^{m}\sum_{j=1}^{k} w(j,i)\log\phi(j) - \sum_{j=1}^{k}\lambda_j\phi(j) + (\sum_{j=1}^{k}\phi(j) - 1)\nu \tag{24}$$

Therefore the KKT conditions (theorem 2) are

1) $\nabla_\phi L(\phi^*, \lambda^*, \nu^*) = 0$
2) $-\phi^*(j) \leq 0 \quad , \quad \forall j$
3) $\sum_j \phi(j) = 1$
4) $\lambda_j^*\phi^*(j) = 0 \quad , \quad \forall j$
5) $\lambda_j^* \geq 0 \quad , \quad \forall i$

After deriving the Lagrangian with respect to a specific $\phi(j)$ we get

$$\phi(j) = \frac{\sum_{i=1}^{m} w(j,i)}{\nu} \tag{25}$$

We do not know ν, so we use the fact that $\sum_j \phi(j) = 1$

$$\sum \phi(j) = \frac{\sum_j \sum_{i=1}^m w(j,i)}{\nu} = 1 \tag{26}$$

Therefore

$$\phi(j) = \frac{\sum_{i=1}^n w(j,i)}{\sum_{i=1}^m \sum_{j=1}^k w(j,i)} \tag{27}$$

$$= \frac{\sum_{i=1}^n w(j,i)}{n} \tag{28}$$

We can see that the KKT conditions holds.

Algorithm:

E-step: for each i, j

$$w(j,i) := P(z_i = j | x_i; \phi, \mu, \Sigma) \tag{29a}$$

$$= \frac{\phi(j) P(x_i | z_i = j; \mu_j, \Sigma_j)}{\sum_{l=1}^k \phi(l) P(x_i | z_i = l; \mu_l, \Sigma_l)} \tag{29b}$$

M-step: for each j

$$\phi(j) := \frac{1}{m} \sum_{i=1}^m w(j,i) \tag{30}$$

$$\mu_j := \frac{\sum_{i=1}^m w(j,i) x_i}{\sum_{i=1}^m w(j,i)} \tag{31}$$

$$\Sigma_j := \frac{\sum_{i=1}^m w(j,i)(x^{(i)} - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^m w(j,i)} \tag{32}$$

Now we must consider when to stop the algorithm. One way is to repeat the two steps until convergence $\theta^{(t+1)} \approx \theta^{(t)}$. Another option is to repeat them for a fixed number of times. Repeating the steps a fixed number of times can prevent over-fitting of the training data set.

F. K-means algorithm

K-means is very similar to EM except that it gives a 'hard' decision for each sample to the centroid to which it belongs. Below we discuss K-means briefly. We are given a training set $\{x^{(1)}, ..., x^{(m)}\}$ and we wish to group it into K different components. Algorithm:

1) Initialize centroids $\mu_1 ... \mu_k \in \mathbb{R}^n$
2) Repeat until convergence: $\{$
    a) for every i, set

$$c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2 \tag{33}$$

    b) for each j, set

$$\mu_j := \frac{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\}} \tag{34}$$

$\}$

To initiate the centroid, we can choose K random training samples. There are other initialization methods. The inner loop assigns each sample to the 'closest' centroid, and moves the centroid to the mean of the points assigned to it.

The K-means algorithm guarantees convergence to a local optima (except in very rare cases, where the K-means oscillate between a few different values), but it does not guarantee convergence to a global optima. A common way to deal with this problem is to run K-means many times with different initiations, and then to pick the one with the lowest distortion. K-means is often used to initiate GMM modelling.

A common use of the K-means algorithm is for initial estimation for GMM parameters. We use the centroids to initiate the Gaussian's μ and set an arbitrary Σ and ϕ.

G. Tips and additional use of GMM

1) Choosing the number of GMM components: [6] Choosing the number of components for modelling the distribution is an important issue. There is a trade-off between choosing too many components which may cause over-fitting (for example, what happens if we use the number of samples?), and choosing too few, which can render a model that

is not flexible enough. Selecting the number of components is crucial for unsupervised segmentation tasks, in which each component represents a separate class. However, when each GMM models a different class and the segmentation is supervised, the selection of the number of components becomes less critical.

2) Full or diagonal covariance?: [5] In the model that is the focus of this lecture, we used unrestricted covariance for the GMMs. The GMM can also be restricted to have diagonal covariance which has two practical advantages:

- There are fewer parameters to be estimated.
- Calculation of the inverse matrix is trivial.

In some applications (e.g., speaker verification), the diagonal model is sufficient. The obvious disadvantage of assuming diagonality, however, is that the different features may in practice be strongly correlated. In this case, the model might be incapable of representing the feature distribution accurately. The figure below illustrates this point.

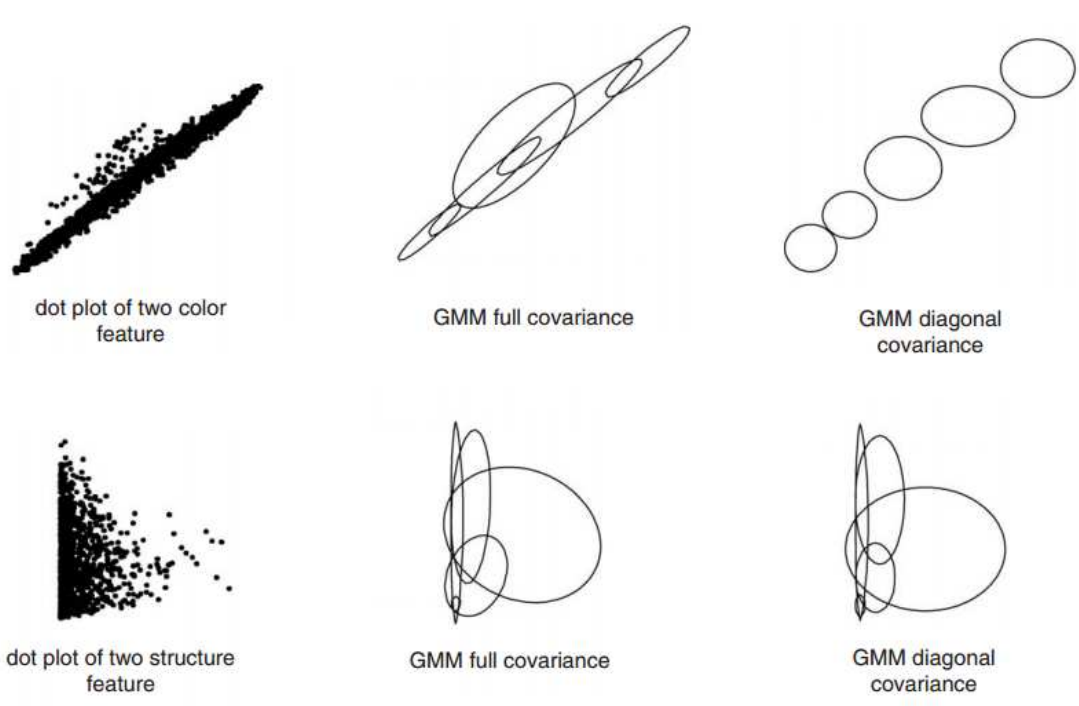

Fig. 2. Modelling the distributions of two different features(first column) using a full covariance(second column) and a diagonal covariance(third column)

On the first row, we see that the features are highly correlated, and therefore the GMM must use a full covariance matrix. On the second row, the difference is far less pronounced, and it seems that a diagonal covariance matrix is sufficient.

3) GMM for unsupervised learning: So far we only talked about using GMM for supervised learning, where each sample has its own label, and a model is built for each class. In many cases, GMM is used for unsupervised clustering. It is useful to gather data from a group into sub-groups that can be modelled well by a gaussian distribution. Using GMM, we can cluster the data into sub-groups that do not have labels or identity. For example, we have a class with n students and we want to split it into three different study groups according to student averages. An addition application of GMM in unsupervised learning is the *Universal Background Model* (UBM). In the UBM, we first generate a general class-independent statistical model by using all samples. We often use the UBM to generate the class-dependent models. This is particularly useful for cases in which we lack a sufficient number of samples of the class we want to identify, but we have many samples of other classes. A good example of this application is in the field of speaker verification, when we may have only a few voice samples of the person we are trying to identify.

APPENDIX

REVIEW: CONVEX FUNCTION AND JENSEN'S INEQUALITY[4]

Definition 1 (convex set) :

A set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$:

$$\theta x + (1-\theta)y \in C. \tag{35}$$

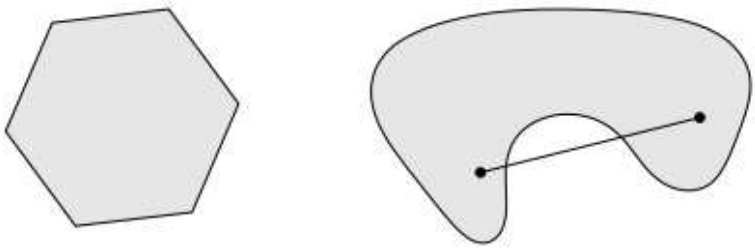

Fig. 3. The hexagon on the left is convex. on the right is a non-convex set.

Definition 2 (convex function) :

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if $\mathrm{dom} f$ is a convex set and if for all $x, y \in \mathrm{dom} f$, and θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \tag{36}$$

It is called strictly convex if

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta) f(y) \tag{37}$$

f is (strictly) concave if $-f$ is (strictly) convex.

Definition 3 (Convex optimization problem) :

The notation

$$\text{minimize} \;\; f_0(x)$$

$$\text{subject to } f_i(x) \leq 0, \quad 1 \leq i \leq m$$
$$h_j(x) = 0, \quad 1 \leq j \leq k$$

describes the problem of finding the x that minimizes $f_0(x)$ among all x that satisfy the constrains. It is called a convex optimization problem if $f_0, ..., f_m$ are convex functions and $h_1, ..., h_k$ are affine.

Definition 4 (Lagrangian) :
For a constrained optimization problem as seen in definition 3 we define the Lagrangian

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{k} \nu_j h_j(x) \tag{38}$$

The idea of the Lagrangian duality is to take the constrains into account by augmenting the objective function with a weighted sum of the constraint functions.

Theorem 1 (Jensen's Inequality) :
Named after mathematician **Johan Jensen**, the theorem relates the value of a convex function of an integral to the integral of the convex function.
In the context of probability theory: if X is a random variable and f is a convex function,

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]. \tag{39}$$

Equality holds if and only if x is constant or f is linear.

Theorem 2 (KKT conditions) :
For an optimization problem with constrains, which we transform to a Lagrangian, let's assume that a strong duality holds (discussion about duality is out of our scope, and can be found in Boyd's book). Then the following condition holds

$$\nabla_x L(x^*, \lambda^*, \nu^*) = 0$$
$$f_i(x^*) \leq 0 \quad , \quad \forall i$$
$$h_j(x^*) = 0 \quad , \quad \forall i$$
$$\lambda_i f_i(x^*) = 0 \quad , \quad \forall i$$

$$\lambda_i^* \geq 0 \quad , \quad \forall i$$

where x^* and (λ^*, ν^*) are a primal and dual optimal points with zero duality gap. This are called *Karush-Kuhn-Tucker* conditions.

You can find a more rigorous discussion of convex optimization in Boyd's book on convex optimization in chapters 1-5[4].

REFERENCES

[1] Andrew NG's machine learning course. Lectures on *Unsupervised Learning, k-means clustering*, *Mixture of Gaussians* and *The EM Algorithm* http://cs229.stanford.edu/materials.html.

[2] Haim Permuter's *Machine leaning course*, lecture 1 http://www.ee.bgu.ac.il/~haimp/ml/lectures.html.

[3] Arthur Dempster, Nan Laird, and Donald Rubin (1977) *Maximum Likelihood from Incomplete Data via the EM Algorithm* https://www.jstor.org/stable/2984875.

[4] Stephan Boyd's *Convex Optimization* https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

[5] Haim Permuter, Joseph Francos, Ian Jermyn, *A study of Gaussian mixture models of color and texture features for image classification and segmentation* 3.4 http://www.ee.bgu.ac.il/~francos/gmm_seg_f.pdf.

[6] Haim Permuter, Joseph Francos, Ian Jermyn, *A study of Gaussian mixture models of color and texture features for image classification and segmentation* 3.5 http://www.ee.bgu.ac.il/~francos/gmm_seg_f.pdf.

[7] Ramesh Sridharan's *Gaussian mixture models and the EM algorithm* https://people.csail.mit.edu/rameshvs/content/gmm-em.pdf.