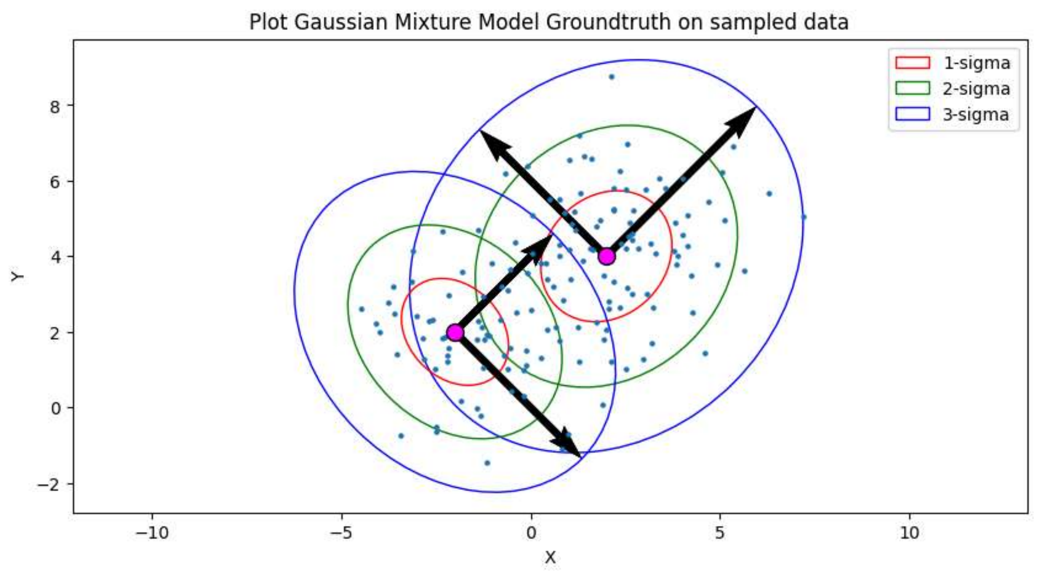
Mixture Discriminant Analysis: A Supervised Mixture of Class-specific Gaussian Mixtures Model

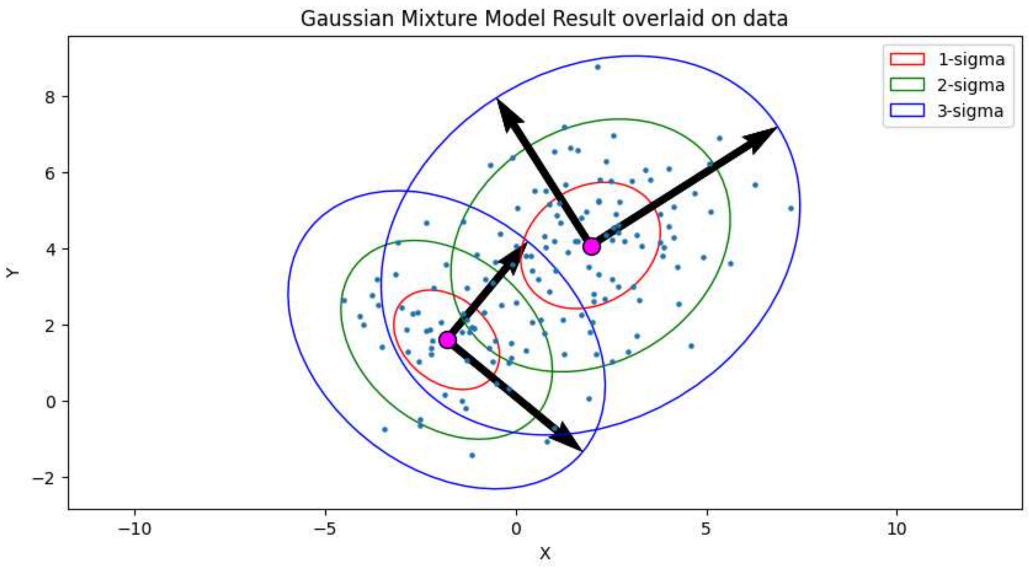
CSDS491 Spring 2024

Quan Le

Inspiration:

The last exercise on assigment 4 confirmed that Gaussian Mixture Models combined with the Expectation-Maximization algorithm can indeed recover the correct clusterings of data if they were synthesized from a mixture of different multivariate normal distributions (Gaussian components). Below are 100 points that are sampled from one bivariate normal distribution and 50 points from another. The EM with Gaussian Mixture Model algorithm was able to recover the original clustering with high accuracy,indicated by the similarity between the bottom and top graph.

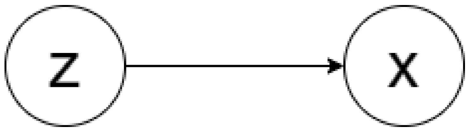




The number of Gaussians were predetermined, as were the parameters (means and covariances). These parameters are then used to sample a mixed pool of data that is going to be run through EM. Since the data pool was synthesized, EM on GMM was able to fit the clusters around the data points perfectly. However, the success of this experiement does not help to showcase the usefulness of the Gaussian Mixture Model in a pratical setting, where the structure of data that needs clustering is most certainly unknown. For me, the moment I bring the concept of a Gaussian Mixture Model into the context of data classification, things become obscured as class information is introduced to the picture. In particular, what is the relation between the ground truth classes and the number of Gaussian components. How do these Gaussian clusters allow us to infer the true class of unseen data points? Why do we even begin with something like a Gaussian Mixture Model for a classfication task and would it be any good if we decided to do so? With this project, I aim to record my explorations into supervised extensions of the Gaussian Mixture Model algorithms for classification, namely QDA and MDA. I will also demonstrate the insights about the overfitting phenomenon that results from using a complex model like MDA to estimate classification boundaries that are far more simple in nature.

Gaussian Mixture Model in an unsupervised context

The traditional Gaussian Mixture Model is an unsupervised learning method for many applications, including pattern recognition. Assume data are independently and identically sampled from some unknown probability density function. A mixture of multivariate normal distributions $\theta_{1:K}$ are used to estimate this density function, based on the result that any smooth density function can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components (Goodfellow et al.,2016). Call the random variable z_k to represent the Gaussian distribution / cluster k^{th} that has a set of parameter θ_k that includes a mean μ_k and a covariance matrix Σ_k.
Our original world in a Gaussian Mixture Model therefore involves only 2 random variables, our data $\mathbf{x}$ and the cluster $\mathbf{z}$, with $\mathbf{x} \in \mathcal{R}^{\mathbf{D}}$.



Say there are N data points. We can use basic Bayes' Theorem to derive the following marginal density function of some point n^{th}:

$$\begin{aligned} p(x_n) &= \sum_{k=1}^{K} p(x_n, z_k) \\ &= \sum_{k=1}^{K} p\left(z_k\right) p(x_n|z_k) \\ &= \sum_{k=1}^{K} \pi_k \, p(x_n|z_k) \end{aligned}$$

Under our Gaussian Mixture Model, the conditional distribution of data over the set of clusters $p(\mathbf{x}|\mathbf{z})$ is defined as follows:

$$p(x_n|z_k) = p(x_n|z_k, \theta_k) = \mathcal{N}_{\mu_k, \Sigma_k}(x_n) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x_s n - \mu_k)'\Sigma^{-1}(x_n - \mu_k)\right)$$

Intuitively, the mixing coefficient π_k tell us the k^{th} cluster's contribution to the overall data distribution - given a data point x_n, how much of the probability density of that point evaluated within cluster k^{th} gets transfer to the overall density of observing that point. It naturally makes sense for pi_k to be in between 0 and 1 and adds up to 1 if we think of them as weight coefficients for the clusters. However, interpret π_k in terms of what probability it expresses, provides a more robust explanation of the constraint. π_k has to be non-negative, and $\sum_{k=1}^{K} \pi_k = 1$ in order to define probability mass function of the discrete cluster random variable z properly. Later, we will see how this pattern of some mixing coeffcients being constrained to express valid probability distributions repeating in the MDA derivation.

Beside the mixing coefficients, the probability of cluster membership (not to be confused with class membership) also plays a central role in parameter estimation. This is simply the posterior probability of the data point x_n belongs to cluster z_k, $p(z_k|x_n)$, or the responsibility of cluster k^{th} to the observation of x_n:

$$
\begin{aligned}
\gamma(z_k, x_n) = p(z_k|x_n) &= \frac{p(z_k, x_n)}{p(x_n)} \\
&= \frac{p(z_k)\, p(x_n|z_k)}{\sum_{k=1}^{K} p(z_k)\, p(x_n|z_k)} \\
&= \frac{\pi_k\, \mathcal{N}_{\mu_k, \Sigma_k}(x_n)}{\sum_{k=1}^{K} \pi_k\, \mathcal{N}_{\mu_k, \Sigma_k}(x_n)}
\end{aligned}
$$

Notably, according to the lecture, the sum of the responbility over all the data points N is the effective number of data points classified to component k.

$$
N_k = \sum_{n=1}^{N} \gamma(z_k, x_n)
$$

At first, just like every other concept in GMM, I tried to understand this summation "probabilistically", hoping it would correspond to some sort of marginalization. Converting each responsibility to its posterior probability interpretation and substitute into the formula, we gained:

$$
\begin{aligned}
N_k &= \sum_{n=1}^{N} \gamma(z_k, x_n) \\
&= \sum_{n=1}^{N} p(z_k|x_n)
\end{aligned}
$$

This is not a marginalization. At this point, I realized N_k cannot be interpreted probabilistically because it is simply not expressing any probability distribution. Instead, it is more appropriate to look at N_k as an effective count. If individual responsibility quantifies how much the k^{th} cluster is explaining a single observed data point x_n, summing over these responsibilities will return an aggregate measure that quantifies how much the k^{th} cluster is explaining (responsible for) the whole data set. As data points are allowed soft/partial assignment to different clusters using probability density (fractional cluster membership), this count π_k is effective and essentially represents the fraction of data explained by cluster k.

This concept of responsibility is also transferrable to MDA. However, its formulation then is no longer simply the posterior probability of data belonging to cluster, but something more complex that involves the extra true class information.

The objective of density estimation is to maximize the log-likelihood of the training data, which is captured by the formula (assuming data is i.i.d):

$$\mathcal{L}(\mathbf{x}) = \log\left(\prod_{i=1}^{N} p(x_i)\right)$$

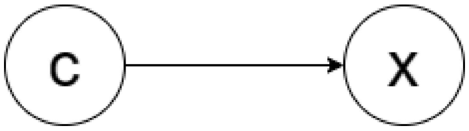
The maximization of this term is achieved by taking the derivative with respect to the model parameters and set it to 0, solving for critical values. The Expectation step estimates the responsibilities, and the Maximization step then recomputes the model parameters based on that estimation. The process is repeated until convergence is reached.

The next section will explain the concept of Mixture Discriminant Analysis - a supervised learning approach extending Gaussian Mixture Model that can be used for classification.

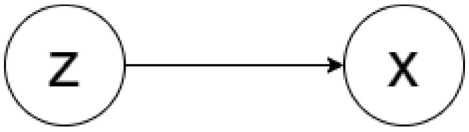
Supervised Mixture of Gaussian Mixtures Model - Mixture Discriminant Analysis Classifier

What is the difference between a supervised and an unsupervised learning approach? It is the presence of some ground truth, or class information that can be leverage to guide the learning process.

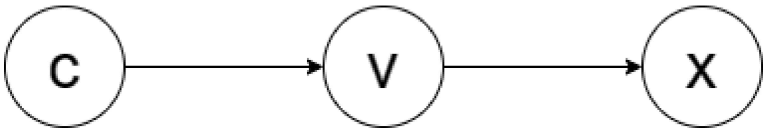
In the training data, we have pairs (x_i, c_i) where each $x_i \in \mathbb{R}^D$ is a feature vector, and $c_i \in \mathcal{J}$ is the corresponding class label, for $i = 1, \ldots, N$. For classification, our primary objective is to maximize the conditional probability density $p(\mathbf{x}|\mathbf{c})$. This density, through Bayes' Rule, allows us to compute the posterior probability $p(\mathbf{c}|\mathbf{x})$ that gives the likelihood of a data point $\mathbf{x}$ belonging to a certain class $\mathbf{c}$. It is this posterior probability that serves as the basis for classifying new instances, thereby assigning them to the class with the highest posterior probability given their observed features.
Without the Gaussian Mixture Modeling, our world involves 2 random variables, $\mathbf{x}$ and $\mathbf{c}$



Compared to our original unsupervised Gaussian Mixture Model:



To incorporate Gaussian Mixture Modeling into this new world that has the class random variable, we have to decide how we want $\mathbf{c}$ to influence $\mathbf{z}$. For me, it was helpful to visualize this modeling decision of merging the two graphs above into one as the new graph below:



But how do we want to model the relationship between $\mathbf{c}$ and $\mathbf{v}$ - the class random variable and the Gaussian distribution latent random variable that we assume to generate x? There are two ways:

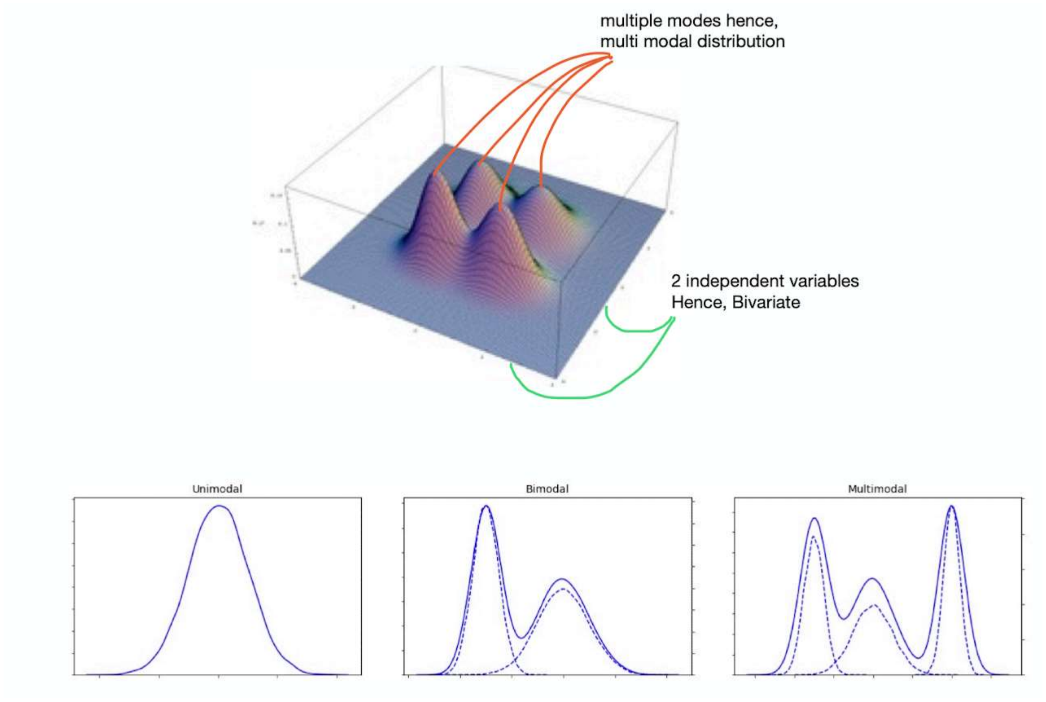
- A direct one-to-one mapping, deterministic relationship: the class label c determines the Gaussian distribution from which the data belonging that class is generated. In other words, this is the simple case in which we model the class after a single Gaussian distribution, essentially collapsing (reducing) $\mathbf{c}$ and $\mathbf{v}$ into one random variable.
- A stochastic relationship: the class label c influences the Gaussian component that generates data belonging to that class through some probability distribution. In this model, each class is modeled as a distinct mixture of Gaussian distributions, thus the class variable $\mathbf{c}$ and the Gaussian latent variable $\mathbf{v}$ stays separated.

This decision is essentially the difference between the two statistical learing generative models Quadratic Discriminant Analysis (QDA) and Mixture Discriminant Analysis (MDA) that aim to predict some most likely class label given an input feature vectors based on training data. A simple analogy to conceptualize the difference between these two models is the bag and the box:

QDA: There's a set of different shaped boxes (Gaussians with distinct covariances), each labeled with a class. When you get a data point from a class, you know exactly which box (Gaussian) it came from.

MDA: There's a set of bags for each class, and inside each bag, there are multiple smaller boxes (Gaussian components) of various shapes. The class label tells you which bag to pick, but there's a probability-driven choice about which small box inside the bag the data point comes from.

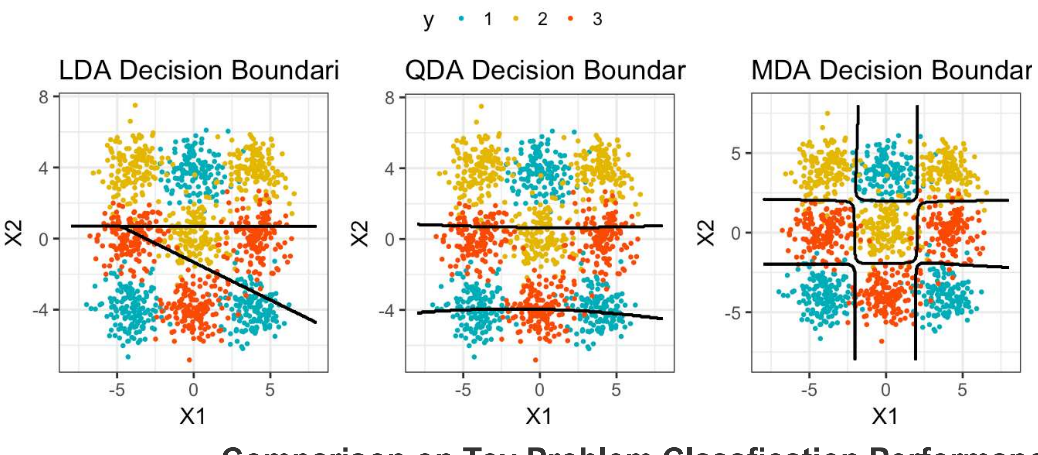
Compared to QDA, MDA creates a complex decision boundary that can take on more irregular shapes due to the assumption that each class distribution is mutlimodal (having multiple local peaks where density is highly concentrated) and thus better represented by multiple Gaussian distributions (subclasses/subpopulations).



A bivariate, multimodal distribution

Despite being more complex, it is always possible for MDA to perform worse than QDA in situations where the assumption of a multimodal class distribution is not necessarily true. With small training dataset, estimating some unimodal class distribution with multiple Gaussians like MDA is an overkill that would lead to overfitting random noises and fluctuations. Therfore, it is essential (as well as difficult) to select a model whose complexity matches the underlying structure of the data.

We can immediately confirm that MDA does outperform LDA and QDA when the assumptions are satisfied through a toy problem. The PCA graph below shows the result.



Comparison on Toy Problem Classfication Performance

Note that in this example, the author is implementing an MDA model that assumes each subpopulation in every class shares a single common covariance matrix. He then runs it on a synthesized data set where that assumption is true, and model outcomes a beautiful result. However, this assumption might not always be applicable in real-world scenarios due to its restrictiveness. Therefore, I attempted to implement an MDA model that relaxes the uniform covariance matrix constraint, allowing distinct shape for each subpopulation of the same class. This would theretically allows more flexibility in fitting the training data at the cost of computational resources and overfitting risk. The Breast Cancer Tumor Classification Dataset is then run through this MDA classifier.

The overall takeaway is that QDA is a Gaussian Mixture Model where class and cluster means the same thing, while MDA is a mixture of class-specific Gaussian mixtures model, where each class is modeled by multiple Gaussian components.The section below demonstrates the derivations of probability formulas that I derived and used during implementation. Through deriving probability density functions and formulate the EM update rule, I learned how powerful fundamental principles like Bayes' Theorem and marginalization as they enable and steer the extension from an unsupervised GMM $\rightarrow$ a supervised MDA classifier.

Derivation of MDA

The derivations of the equations below are elaborated upon the journal article Discriminant Analysis by Gaussian Mixtures (Hastie and Tibshirani, 1996)

The training data consists of $(x_k, c_k) \in R^D, \mathcal{I}, k = 1, \ldots, N$. We divide each class $c_i, i = 1, \ldots, I$ into R_i Gaussian components from which data points are sampled, each denoted by $z_{ij}, j = 1, \ldots, R_i$.

The model assumes every component has its own mean μ_{ik} and covariance matrix Σ_{ik} (note that this is different from the toy problem model above, where a common covariance is shared between all subclasses).

Now our world has 3 random variables:

- $\mathbf{x}$: data points
- $\mathbf{c}$: class
- $\mathbf{z}$: subclasses (Gaussian clusters)

In a supervised learning context, the goal is to estimate a set of parameters that maximizes the likelihood of the training data conditioned on their class labels $p(\mathbf{x}|\mathbf{c})$. This is different from the unsupervised approach, where the **marginal likelihood** of data is maximized, $p(\mathbf{x})$.

Consider marginalizing over all the latent subclass variable $\mathbf{z}$ to get the conditional probability density of data point x_k given its class c_i

$$\begin{aligned} p(x_k|c_i) &= \sum_{j=1}^{R_i} p(z_{ij}, x_k|c_i) \\ &= \sum_{j=1}^{R_i} p(z_{ij}|c_i)\, p(x_k|z_{ij}, c_i) \quad \text{(Bayes' Rule)} \\ &= \sum_{j=1}^{R_i} \pi_{ij}\, \mathcal{N}_{ij}(x_k) \end{aligned}$$

The conditional log-likelihood of the data is:

$$\log\left(\mathcal{L}(\mathbf{x}|\mathbf{c})\right) = \log\left(\prod_{k=1}^{N} p(x_i|c_{g_i})\right) = \sum_{k=1}^{N} \log(p(x_i|c_{g_i}))$$

with g_i denotes the index of the class of the i^{th} data point ($1 \leq g_i \leq I$)

This process of using Bayes' Rule and marginalization to compute the likelihood of data was done in the same exact way as the unsupervised GMM. The only difference is the conditioning on the class variable $\mathbf{c}$ and a tweak to indexing the Gaussian components z_{ij} - which is two-dimensional since now we look at each cluster as the j^{th} subclass of the i^{th} class. Just like in the original model, looking at a mixing coefficient as a probability gives me a very convenient and intuitive way to grasp what it means. A mixing coefficient π_{ij} is simply the posterior probability $p(z_{ij}|c_i)$ - how much does the j^{th} components of class c_i is explaining for the observation of data from that class.

To maximize this conditional log-likehood, we do the standard approach of setting the partial derivative with respect to μ_{ij} and Σ_{ij} of $\log\left(\mathcal{L}(\mathbf{x}|\mathbf{c})\right)$ equal to 0, then solve for values μ_{ij} and Σ_{ij} respectively.

$$0 = \frac{\partial \log\left(\mathcal{L}(\mathbf{x}|\mathbf{c})\right)}{\partial \mu_{ij}} = -\sum_{g_k=i} \left[\frac{\pi_{ij}\mathcal{N}_{ij}(x_k)}{\sum_{r=1}^{R_i} \pi_{ir}\, \mathcal{N}_{ir}(x_k)}\right](x_k - \mu_{ij})$$

The term within the square bracket has the same "responsibility" interpretation as $\gamma(z_{nk}) = p(z_k|x_n)$ in the unsupervised version. It is essentially $\gamma(z_{kij}) = p(z_{ij}|x_k, c_i)$, or the probability the k^{th} instance belongs to the j^{th} Gaussian component of the i^{th} class.

Notation $\sum_{g_k=i}$ means to sum over all instances belonging to the i^{th} class. The same analogy applies to $\sum_{g_k=i} \gamma(z_{kij})$ here as $\sum_N \gamma(z_{nk})$ in the unsupervised derivation. $N_{ij} = \sum_{g_k=i} \gamma(z_{kij})$ is the effective count of data points of class i^{th} that get assigned to cluster j^{th}

Solve for μ_{ij}, we get:

$$\mu_{ij} = \frac{1}{N_{ij}} \sum_{g_k=i} \gamma(z_{kij}) x_k$$

Similarly, set $0 = \frac{\partial \log(\mathcal{L}(\mathbf{x}|\mathbf{c}))}{\partial \Sigma_{ij}}$ and solve for Σ_{ij}:

$$\Sigma_{ij} = \frac{1}{N_{ij}} \sum_{g_k=i} \gamma_{kij} (x_k - \mu_{ij})(x_k - \mu_{ij})^T$$

To solve for π_{ik}, we have to use Lagrange multipliers to introduce the constraint for the mixing coefficients to define a valid discrete probability distribution for $p(z_{ij}|c_i)$: $\sum_{j=1}^{R_i} \pi_{ij} = 1$, for each class, we need a seperate Lagrangian with a Lagrange multiplier and a log-likelihood that is conditioned on the class:

$$\log(\mathcal{L}(\mathbf{x}|c_i)) + \lambda_i (\sum_{j=1}^{R_i} \pi_{ij} - 1)$$

Repeat the approach used for mean and covariance, set partial derivative of the Lagrangian with respect to π_{ij} equal to 0 and solve for π_{ij}:

$$0 = \sum_{g_k=i} \frac{\mathcal{N}_{ij}(x_k)}{\sum_{r=1}^{R_i} \pi_{ir} \mathcal{N}_{ir}(x_k)} + \lambda_i \quad (1)$$

Multiply both sides by π_{ij} for every possible value of $j = 1, \ldots, R_i$ and add them together, using the sum-to-one constraint, we have:

$$\begin{aligned} 0 &= \sum_{g_k=i} \frac{\sum_{j=1}^{R_i} \pi_{ij} \mathcal{N}_{ij}(x_k)}{\sum_{r=1}^{R_i} \pi_{ir} \mathcal{N}_{ir}(x_k)} + \lambda_i (\sum_{j=1}^{R_i} \pi_{ij}) \\ \Leftrightarrow 0 &= \sum_{g_k=i} (1) + \lambda_i (1) \\ \Leftrightarrow -N_i &= \lambda_i \end{aligned}$$

$\Rightarrow$ The Lagrange multiplier for class i^{th} is essentially the number of data points belonging to the class

With $\lambda_i = -N_i$, substitute back into (1) multiplied by π_{ij} for both sides, we can solve for π_{ij}:

$$\begin{aligned} N_i \pi_{ij} &= \sum_{g_k=i} \frac{\pi_{ij} \mathcal{N}_{ij}(x_k)}{\sum_{r=1}^{R_i} \pi_{ir} \mathcal{N}_{ir}(x_k)} = \sum_{g_k=i} \gamma(z_{kij}) = N_{ij} \\ \Rightarrow \pi_{ij} &= \frac{N_{ij}}{N_i} \end{aligned}$$

This derived expression for the mixing coefficient π_{ij} intuitively makes sense - it quantifies the proportion of N_i observed data points from class i^{th} that can be accounted for by the j^{th} subclass.

With the complete update formula, we can apply the Expectation Maximization iteratively until convergence.

The classification task can be achieved by computing the class-posterior $p(c_i|x_{new}) = p(c_i)p(x_{new}|c_i) = p(c_i)\sum_{j=1}^{R_i}\pi_{ij}\mathcal{N}_{ij}(x_{new})$ for each class, and assigned x_{new} to the class with the highest probability.

The section below will discuss and evaluate the results of running an MDA classifier on an empirical dataset.

Data and Implementation

Data Overview

The dataset chosen is the Breast Cancer Wisconsin (Diagnostic) Data set, which includes 569 feature vectors that are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, each describe the characteristics of the cell nuclei present in the image. The groud truth label is whether the mass is benign or malignant, thus the classification task is binary.
There are 10 real-valued features computed for each cell nucleus image, of which the mean, standard error, and mean of the 3 largest values are recorded, resulting in 30 real-valued features. The data set is complete.

I chose this data set to experiment on because of initial concerns about skewed class distribution that can result in numerically unstable probability densities. As my primary objective is to successfully implement MDA, I wanted to avoid the issue of singular covariance matrices. The data set has a fairly balanced class distribution (357 benign and 212 malignant), plus each class has sufficent data points so that the covariance matrices will behave stably as they get updated.

0 mean and unit variance standardization is applied. Data is split into an 80/20 train test set (reasons discussed in implementation)

Implementation

The model first splits the data into bags of instances that belong to the same class. To obtain the initial model parameters, k-means is run on each bag. The cluster centroids are then used as initial μ_{ij}, the covariance of each cluster is estimated by a diagonal matrix Σ_{ij} based on the k-means cluster assigment, and mixing coefficients π_{ij} is initialized to $1/R_i$ for class i. Note that for ease of implementation, the number of clusters within each class is set to be equal to each other, meaning $R_i = Rj = K$, $\forall 1 \leq i, j \leq J$ and K is treated as a hyperparameter chosen prior to training.

Hastie and Tibshirani (1996) noted that there is no automatic way for selecting a good value of R_i althought it is possible to be surveyed through a validation set.

K is the critical parameter for MDA because it is the model's assumption about the true shape of the data distribution. If the true distribution is indeed multimodal (having multiple peaks), using MDA with large value of K can be beneficial as it allows a more fine-grained estimation of data. If the data distribution is either unimodal or the number of modes is significantly less than K, the model with K number of clusters representing each class would overfit the noises within the training data and generalize badly to unseen instances. On the other hand, if the true distribution looks like a gru with lots of peaks (visualize the bivariate image in the third section), using one single inverted cone to fit over the top those peaks - fitting just one Gaussian component for each $p(\mathbf{x}|\mathbf{c_i})$ - will most likely underfit. For me, underfit and overfit in density estimation has a lot to do with that mental image of putting a cloth over some mountain range (training data) and use the recorded shape $[\text{estimated}\, p(\mathbf{x}|\mathbf{c_i})\text{ from data}]$ to predict the shape of some other mountain range (test data). Now, should the cloth be wet and hug very tightly over these peaks (overfit), or should it be dried and retain some inherent degree of firmness, perhaps because the underlying structure representative of all mountain ranges is not as peaky as the one that we put our cloth over? The cloth here is our MDA classifier and its parameters, and K decides whether it is a wet or dry cloth.

Through implementing a MDA classifier and fit it to a training data set and evaluate the performance on both the train and test set for different values of K, I aim to record this overfitting phenomenon. As the model becomes more complex (increasing K), accuracy on the train set should increase while on the test set it should decrease. Observing this result will confirm that my cloth-mountain analogy is appropriate and indeed true. Note that when $K = 1$, the classifier is QDA because it models $p(\mathbf{x}|\mathbf{c_i})$ for some class i by a single Gaussian distribution, and when $K > 1$, the classifier is MDA (A mixture of class-specific mixtures model).

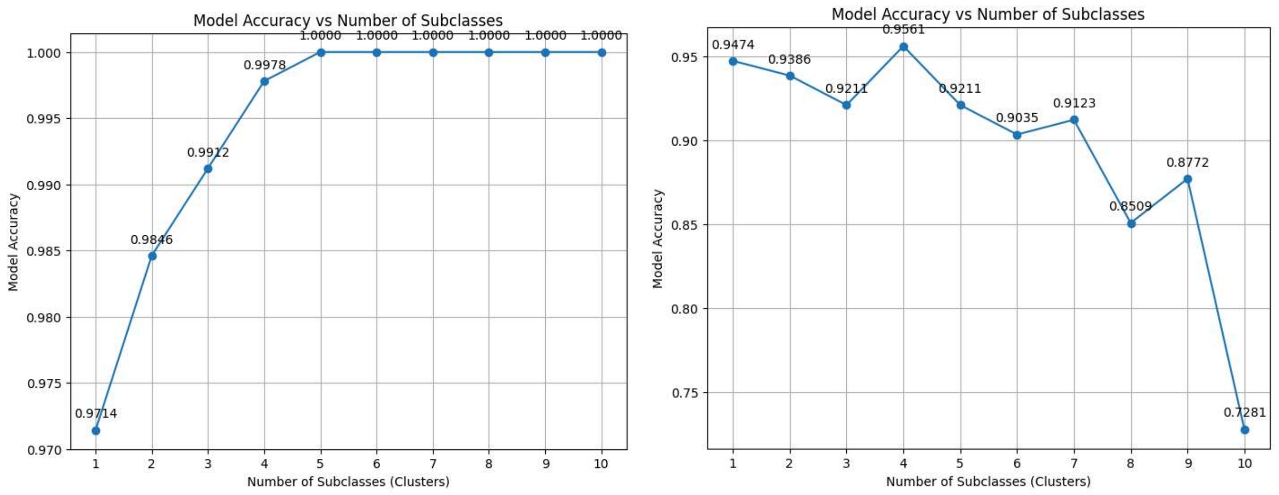
The EM update rule is as derived above:

$$\mu_{ij} = \frac{1}{N_{ij}} \sum_{g_k=i} \gamma(z_{kij}) x_k$$

$$\Sigma_{ij} = \frac{1}{N_{ij}} \sum_{g_k=i} \gamma_{kij}(x_k - \mu_{ij})(x_k - \mu_{ij})^T$$

$$\pi_{ij} = \frac{N_{ij}}{N_i} \text{ With } \gamma(z_{kij}) = \frac{\pi_{ij}\mathcal{N}_{ij}(x_k)}{\sum_{r=1}^{R_i} \pi_{ir}\, \mathcal{N}_{ir}(x_k)} \text{ and } N_{ij} = \sum_{g_k=i} \gamma(z_{kij})$$

Result and Evaluation:

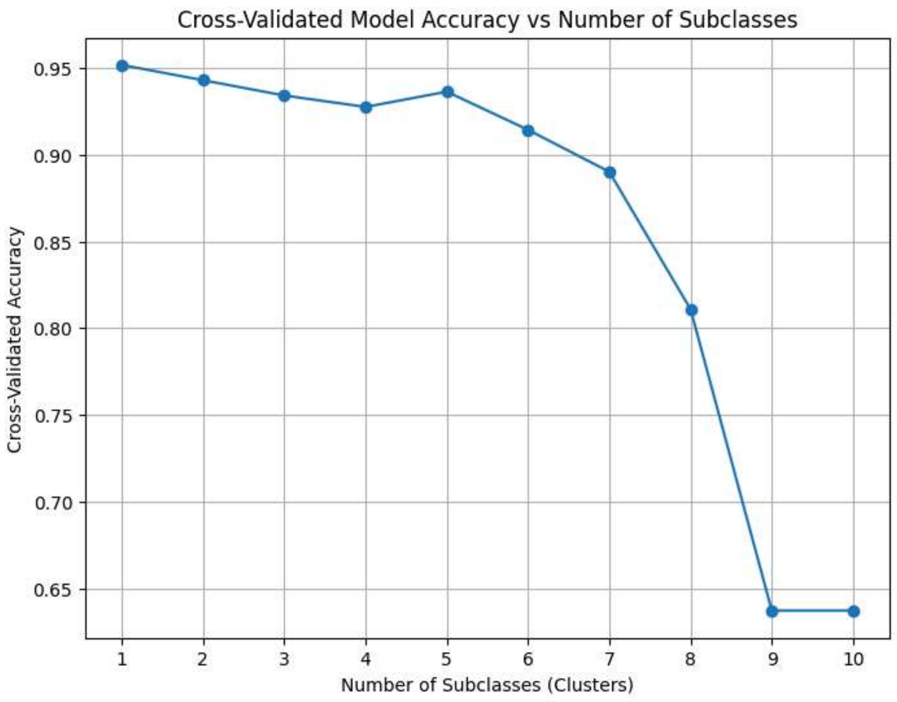


Performance as a function of K on Train Set (left) and Test Set (right)

The graph is a total success given the purpose of portraying the expected overfitting phenomenon. As K increases, on the left graph of the training set, accuracy increases monotonically as the class conditional distribution is approximated with more precision and granularity. As a result, the estimated distribution became too intricate to generalize well to unseen test data. Again, I am going to deploy the cloth-mountain analogy. On the train set, increasing K leads to higher accuracy due to the wet cloth hugging our training mountain range better and better. However, using the shape of this very soft cloth to predict some other mountain range that is not as peaky will not do any good. Therefore, accuracy drops as K increases on the test set. Interestingly enough, this relationship is non-monotonic, as there are larger K values that actually outperforms the lower ($K = 4$). To my interpretation, the K values that corresponds to unusual rise in accuracy are the model concepts that align with the underlying structure of the class conditional distribution $p(\mathbf{x}|\mathbf{c})$ just right to capture the representative patterns without being overly sensitive to noises.

From the performance graph, we've also seen that MDA can outperform QDA once the assumption of the model about the distribution shape aligns with reality. This is evident by $K = 4$ giving the best accuracy value 0.9561 on the test set compared to the second-higest 0.9474 from $K = 1$. However, be cautious not to assert that 4 is the best choice for the model hyperparameter K on this data as the improvement can simply come from sample variability rather than a genuine superiority of the model. To ensure the robustness of model assessment, we can employ techniques like data augmentation and cross-validation to determine if $K = 4$ does indeed generalizes the best.

With five-fold cross-validation and average accuracy metrics, the performance graph is as below:



Performance as a function of K with CV on Test Set

Performance graph with cross-validation enforced clearly demonstrated that the improvement in accuracy observed from $K = 1$ to $K = 4$ purely arised from data variance. It is not indicative of the $K = 4$ model being more generalizable model as it will be outperfoedm by the $K = 1$ model on average. Therefore, MDA is perhaps an overly complex model for this data set and QDA would be a better choice.

Conclusion and Future Direction:

A supervised approach to Gaussian Mixture Model is QDA, where the class-conditional distribution $p(\mathbf{x}|\mathbf{c_i})$ for class i^{th} is assumed to be a Multivariate Normal Distribution with distinct mean and covariance matrix. The MAP framework is then invoked to classify a new instance x_{new} to the class $i = argmax(p(c_i|x_{new}))$ $(\mathbf{x}|\mathbf{c_i})$.

Within the framework of Maximum Likelihood Estimation classifiers, MDA is a natural extension of QDA that models the class-conditional distribution by a mixture of Gaussian components instead of a single Gaussian. This extension is natural in a sense that simple Bayes Rules can be used to derive necessary probability expression and update rules for MDA from QDA, as demonstrated in section 3.

Theoretically, as the number of Gaussians used to estimate each class increase, MDA should be able to capture decision boundary of higher complexity than quadratic - which QDA is incapable of, thus improving accuracy in such cases. However, for simple decision boundaries, MDA is an overkill as it's prone to overfit. Backing off to QDA then will the better choice as it generalizes better, a fact that was confirmed by running the classifier on the Breast Cancer Diagnostic Data set. Therefore, my final answer to the original research problem of whether Gaussian Mixture Model can be extended to work in a supervised approach, is yes. However, one has to be very careful with their model selection. If the model's assumptions does not align well with true underlying distributions (overly complex or overly simple), then overfit would surely happen - tradeoff between model complexity and generalization. It isn't clear how we can avoid overfitting without running the selected model on some validation set, but this belongs to a class of more general, persistent concern of machine learning. The true data distribution is unknown, and we are estimating it to the best of our ability using seen examples.

Along with realizations above, this project also gained me valuable insights about the process of probabilistic modeling. In particular, how random variable can be inserted in and out of a density function arbitrarily using simple Bayes' Rules. This flexibility allows us to not only conveniently convert an unsupervised Gaussian Mixture Model to a Supervised one (QDA), but even a "better" version MDA (in certain scenarios). I was surprised to learn that there was no library in Python that implements the Mixture Discriminant Analysis classifiers and was content to successfully implement one.

Future direction of research into Mixture Discriminant Analysis involves whether it is possible and beneficial to flexibly model different class by varied numbers of Gaussian components R_i. A little even more far fectched is to come up with some heuristics to guide the update of this R_i overtime, either increase or decrease, making the model essentially adaptive to better capture the inhererent separability of data. Hastie and Tibshirani (1996) also discussed a dimensionality reduction method involving multiple linear regressions followed by eigen-analysis that serves to reduce computational requirements in MDA.

References

Hastie, T., & Tibshirani, R. (1996). Discriminant Analysis by Gaussian Mixtures. *Journal of the Royal Statistical Society. Series B (Methodological), 58*(1), 155–176.

Ma, J., Gao, W. The supervised learning Gaussian mixture model. *J. of Comput. Sci. & Technol. 13*, 471–474 (1998). https://doi.org/10.1007/BF02948506

Dr. Michael Lewicki’s lecture slides

John Ramey: A Brief Look at Mixture Discriminant Analysis | R-bloggers