EM Algorithm
Using K-Means for GMM

2025-1 Machine Learning HW#3

학번: 2021202087

이름: 장현웅

1. 과제 제목 및 목적

본 과제의 목표는 Iris 데이터셋(150개 샘플 * 4개 feature) 에 대해 Expectation-Maximization (EM) 알고리즘을 통해 train된 Gaussian Mixture Model (GMM) 을 사용해 클러스터링 알고리즘을 구현하고, 파라미터를 학습한 뒤, 비지도 클러스터링결과를 K-means와 비교 분석하는 것이다. 초기화를 위해 랜덤 값을 대입하는 대신, K-Means 클러스터링 파라미터를 초기값으로 지정하였다.

GMM은 각각의 클러스터가 다변량 Gaussian 분포를 따를 것으로 가정하고, EM은 잠재 변수가 존재하는 Maximum likelihood estimation 과정에서 직접적으로 최적화하기 어려운 우도 함수를 반복적으로 최적화하기 위해 사용된다.

본 과제의 목적은 EM 알고리즘의 수식을 이해하고, 응용 과정까지 정리된 sudo-code를 보고 실제 코드 상에서 구현해볼 수 있도록 하는 것이다.

2. 소스 코드에 대한 설명

HWJ.py 파일은 크게 다음과 같이 구성된다.

1. 라이브러리 & 시드 설정
2. 클래스 EM 정의
3. plotting() 함수 정의
4. main 블록

EM 클래스 내부에는 6개의 메서드가 구현되어 있으며 Iris 데이터를 기반으로 GMM using EM algorithm을 수행한 후 K-means와 비교한다.

2.1. initialization(...)

입력 데이터 X (N = 150 * D = 4) 에 대해, K means를 먼저 수행하여, 각 클러스터 중심 μ_k를 초기값으로 가져오고, 각 샘플 x_n의 라벨을 얻는다. (test용)

mean 파라미터는 k-means 파라미터의 중심으로 계산되고, 각 클러스터 k에 속한 샘플 X_k에서 Σ_k = cov(X_k)를 계산한 후, 매우 작은 값을 더해 정규화하여 siga 파라미터에 저장한다.

전체 데이터 수로 각 클러스터에 속한 데이터 수를 나눈 값을 pi 파라미터에 저장하고, 추후 EM 알고리즘을 통해 매번 업데이트될 값인 gamma 파라미터는 0행렬로 초기화한다.

완전히 랜덤한 값으로 초기화시 local optimum에 빠질 수 있기 때문에, K-mean를 활용해 초기화하였다.

2.2. multivariate_gaussian_distribution(...)

다변량 정규분포(MVN)의 PDF값을 각 샘플에 대해 계산해, likelihood 벡터를 반환한다.

$$\mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma)}} \exp\left[-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right].$$

여기서 determinant를 계산하는 부분은 numpy의 linalg.det(), linalg.inv()를 활용해 계산했다. 간단한 수식적 대입이므로, 구체적 설명은 생략하도록 한다.

2.3. expectation(...)

EM 알고리즘의 E-step을 수행한다.

각 클러스터 k마다 multivariate_gaussian_distribution()을 호출해, N(x_n |, parameters) 벡터를 얻는다. 이후 사전확률 * likelihood를 곱해 분자를 계산하고, normalize한다.

이는 maximum log likelihood 과정에서, 오목함수가 되는 L(theta)의 하한선을 젠센부등식을 통해 정의하고, 그 과정에서 도입한 gamma(z_nk) (아래 자료에서는 z(x))의 도출된 수식의 결과이다.

$$\begin{aligned} L(\theta) &\triangleq \log f(y;\theta) \\ &= \log \int_x q(x,y;\theta)\,dx \\ &= \log \left[\int_x z(x) \frac{q(x,y;\theta)}{z(x)}\right] dx \\ &\geq \int_x z(x) \log\left[\frac{q(x,y;\theta)}{z(x)}\right] dx \\ &\triangleq F(z,\theta) \end{aligned}$$

Latent variable `x' is introduced.
(missing variable!!!)
Arbitary pdf of `x',

Jensen's inequality

Equation of **EXPECTATION**
cf. $E[W(x)]=\int_x W(x)z(x)dx$,
$(W(x) = \log\left[\frac{q(x,y;\theta)}{z(x)}\right]$,
y and Θ are constant)

1) Calculate $z^k(x)$

Let us now define

$$w(x|y;\theta^k) \triangleq \frac{q(x,y;\theta^k)}{f(y;\theta^k)} = \frac{q(x,y;\theta^k)}{\int_x q(x,y;\theta^k)\,dx}$$

then note that

$$\begin{aligned} F(z,\theta^k) &\triangleq \int_x z(x) \log\left[\frac{w(x|y;\theta^k) f(y;\theta^k)}{z(x)}\right] dx \\ &= \log f(y;\theta^k) - D(z(x)\|w(x|y;\theta^k)) \end{aligned}$$

where

$$D(z_1(x)\|z_2(x)) \triangleq \int_x z_1(x) \log\left[\frac{z_1(x)}{z_2(x)}\right] dx$$

(D is the Kullback-Leibler distance)

In order to maximize $F(z,\Theta^k)$ with respect to $z^k(x)$,

$$z^k(x) = w(x|y;\Theta^k) = P(X|Y,\Theta^k),$$

a priori = a posteriori

위 수식에서 omega는 코드에서의 gamma이다. KL-distance가 0이 되어야만 하한선이 최대

화되므로, gamma에 사후 확률의 정의식을 사용해 계산하면 됨을 알 수 있다. 이렇게 정의한 EM 알고리즘은 매 step마다 원래의 로그우도를 항상 단조 증가시킨다. (하지만, 전역 최댓값까지 가는 것은 보장하지 않는다.)

pdf_matrix를 만들어서 MVN 값을 계산한 뒤, weighted_pdfs로 사후확률의 분자를 계산한다. 이후 denominator를 모든 weighted_pdfs 값 합으로 계산해서, gamma 값을 계산한다.

2.4. maximization(...)

도출한 gamma (latent variable이자 사후확률) 값을 대입하여, 평균, 공분산, prior 파라미터를 업데이트한다. 수식은 다음과 같이 첨부한다.

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_{n,k}\, x_n}{N_k} = \frac{(\gamma_{:,k})^T X}{N_k}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_{n,k}\,(x_n - \mu_k)(x_n - \mu_k)^T}{N_k} + 10^{-6} I$$

$$\pi_k = \frac{N_k}{N}.$$

2.5. fit(...)

self.initialization(...)을 호출한 후, 최대 self.iteration 번 EM 루프를 반복한다. 이전 감마 값을 prev_gamma에 저장해둔 뒤, E-step 이후 gamma를 갱신하고, 만약 이전과 이후 값이 1e-4 이내로 차이가 난다면 iteration 완료 전 종료되게 하였다.

그렇지 않다면, prev_gamma에 deep copy로 새로운 gamma를 저장 후, M-step으로 파라미터를 갱신한다.

반복 종료 후, 가장 높은 감마 값을 가지는 레이블을 머신러닝 결과로 할당한다.

2.6. plotting(...)

seaborn.pairplot(...,diag_kind = 'kde')로 연속확률변수에 대해 격자 시각화를 진행했다. 세 이미지를 em, original, kmeans .png로 저장 후, 마지막 kmeans에 대해 이미지 생성 시 세 이미지를 하나로 합쳐 추가로 보여주도록 하였다.

3. 실행 결과 및 분석

3.1. 10회 Run 결과

아래와 같이, 시드를 랜덤하게 하여 10번 반복한 결과를 첨부한다.

seed = 0

```
pi :            [0.2992042  0.33333333 0.36746247]
count / total : [0.3        0.33333333 0.36666667]
EM Accuracy: 0.97    Hit: 145 / 150
KM Accuracy: 0.89    Hit: 134 / 150
계속하려면 아무 키나 누르십시오
```

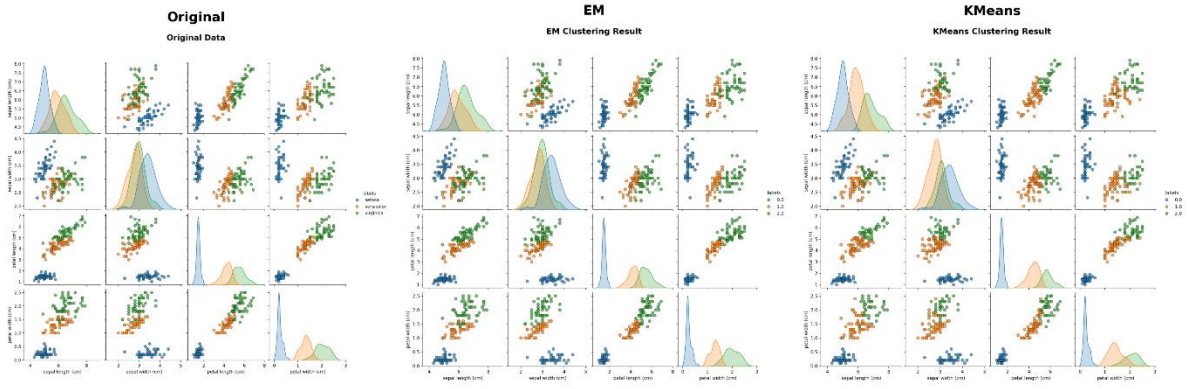
seed_num = 40000

```
pi :            [0.3674628  0.29920387 0.33333333]
count / total : [0.36666667 0.3        0.33333333]
EM Accuracy: 0.97    Hit: 145 / 150
KM Accuracy: 0.89    Hit: 134 / 150
계속하려면 아무 키나 누르십시오
```

80000

```
pi :            [0.33333333 0.29920387 0.3674628 ]
count / total : [0.33333333 0.3        0.36666667]
EM Accuracy: 0.97    Hit: 145 / 150
KM Accuracy: 0.89    Hit: 134 / 150
```

3.2. 주어진 시드에 대한 플롯



다음과 같이 첨부한다.

3.3. 왜 pi 값과 count/total이 유사한 값으로 나타나는가?

실제 M-step에서 pk = Nk/N으로 직접 업데이트되며, 수렴하는 시점에는 대부분 gamma 분포가 {0,1}로 확률이 각 데이터 분포에 대해 원 핫 인코딩에 유사하게 바뀌기 때문이다.