BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO 03

Equipe técnica:

Bruno Leonardo, Clebson Santos, Victor Guedes, e Wallan Melo

Trabalho como parte de obtenção de nota para a disciplina de Sistemas de Apoio à Decisão.

Orientador: Prof. Helder Seixas Lima

Januária MG

2025

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO 03: Seminário em Algoritmos de Agrupamento

Januária MG

2025

Trabalho 3 – Seminário em Algoritmos de Agrupamento

SUMÁRIO

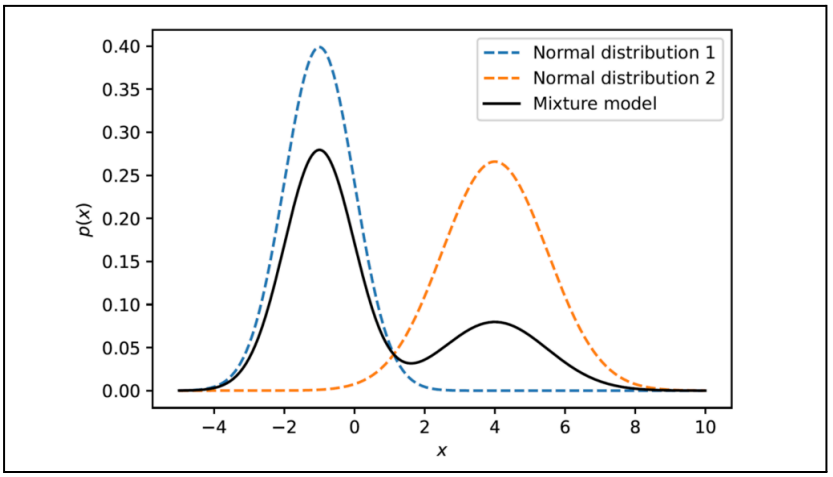
Objetivo

Estudar diferentes algoritmos de agrupamento e desenvolver habilidade de apresentação em grupo no formato de seminário.

Gaussian Mixture Model

1. Explicação do funcionamento do algoritmos

O Modelo de Misturas Gaussianas, ou GMMs, é uma técnica de aprendizado não supervisionado, que representa os dados como uma mistura de várias distribuições gaussianas, e é vista como uma generalização do k-means. Cada cluster é uma Gaussiana com média, variância, e peso.



As linhas tracejadas representam as distribuições normais individuais, e a linha preta contínua mostra a mistura resultante. Este gráfico ilustra como o modelo de mistura combina as duas distribuições, cada uma com sua própria média, desvio padrão e peso na mistura geral.

1.1 Principal Diferença entre GMM e K-Means

O K-Means atribui cada ponto de dados ao cluster de seus vizinhos mais próximos, já o gmm introduz uma abordagem probabilística ao modelar pontos de dados como uma mistura de múltiplas distribuições gaussianas.

Diferença entre GMM e K-Means:

Modelo de Mistura Gaussiana	K-Means
Mais versátil, mas também mais complicado de treinar.	Não serve para muitos propósitos, mas é simples o suficiente para treinar.
Alto requisito de tempo de execução	Treinar mais rápido e com menor tempo de corrida.
Parte-se do pressuposto de que cada ponto de dados se origina de uma combinação de distribuições gaussianas.	Não faz suposições. Simplesmente divide os dados em clusters.
Leva em consideração a variância	Não aborda a variância de forma alguma.
Mais eficazes, pois conseguem lidar com valores ausentes.	Não é possível lidar com dados faltantes, portanto, serão necessários recursos para limpar ou complementar os dados.
O formato dos aglomerados é flexível e pode ser alterado.	Limitado a aglomerados esféricos
Mais preciso para conjuntos de dados pequenos e clusters que não são distintos.	Mais preciso quando o conjunto de dados é grande e possui agrupamentos distintos.

1.2 Como Funciona o GMM:

No GMM cada componente é uma distribuição Gaussiana, um modelo de mistura gaussiana é um modelo de mistura comum, onde a densidade de probabilidade é dada por uma mistura de distribuições gaussianas:

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$$

Um modelo de mistura gaussiana

Onde:

- p (x) é a função de densidade ou massa geral do modelo de mistura.
- K é o número de distribuições de componentes na mistura.
- w□ é o peso de mistura do k -ésimo componente, com 0 ≤ w□ ≤ 1 e a soma dos pesos sendo 1. w□ também é conhecido como a probabilidade a priori do componente k, esse pso determina a influência que cada gaussiana individual possui na mistura final.

- x é o vetor de dimensão d, um vetor de dados de entrada.

- μ□ é o centro do agrupamento, em uma distribuição unidimensional, este será um vetor com somente um único valor, mas em uma distribuição n-dimensional, será um vetor com n valores.

- Σ□ esta é a dispersão/formato da própria Gaussiana. Em uma distribuição unidimensional, será um único valor, mas em uma distribuição n-dimensional, será uma matriz nxn.

- N (x ; μ□ , Σ □) é a função de densidade normal multivariada para o k -ésimo componente:

$$\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right)$$

1.3 MULTIVARIADA

1 - Nessa primeira parte:

$$\frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}}$$

É uma constante de normalização, com a função de garantir que a área total abaixo da curva seja 1, e vai depender diretamente da dimensão d, pois quanto maior a dimensão maior o termo.

2 - Nessa segunda parte:

$$\exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right)$$

Essa é a parte mais importante do modelo de mistura gaussiana multivariada, isso porque ela determina como a densidade diminui conforme o ponto se afasta do centro do cluster.

$-\frac{1}{2}$ Esse multiplicador serve para ajustar a distância para uma escala apropriada para o modelo.

$(\mathbf{x} - \mu_k)$ É um vetor que aponta do centro do cluster até o ponto x, medindo o deslocamento do ponto.

Σ_k^{-1} É a inversão da matriz de covariância, ela mede a dispersão ao contrário, se um determinado ponto está longe na direção de menor variância, então a sua penalização será maior.

1.4 UNIVARIADA

No caso de distribuição gaussianas univariadas, a densidade de probabilidade pode ser simplificada para:

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x; \mu_k, \sigma_k)$$

Um modelo de mistura de distribuições gaussianas univariadas

Onde:

- μ□ é a média do k -ésimo componente gaussiano.
- σ□ é a variância do k -ésimo componente gaussiano.
- N (x ; μ□ , σ□) é a função de densidade normal univariada para o k -ésimo componente:

$$\mathcal{N}(x; \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

1 - Nessa Primeira parte:

$\frac{1}{\sqrt{2\pi}\sigma_k}$ É o fator normalização, ele é o termo responsável por garantir que a área sob a curva seja sempre 1.

2 - Nessa Segunda Parte:

$\exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right)$ Esse termo é o responsável pelo formato de sino que o modelo gaussiana tem, gerando uma curva simétrica com o seu ponto máximo sendo a média.

$(x-\mu_k)^2$ Mede o quão longe o ponto x está do centro, é quanto maior a distância, a penalização no expoente será aumentada, o que irá diminuir a probabilidade.

$2\sigma_k^2$ Controla o peso da distância, por exemplo, se o cluster é muito variado, ou seja com o **σ** grande, os pontos distantes não sofrem uma penalização tão forte. Já se for pouco variado, ou seja com o **σ** pequeno, até as pequenas distâncias irão diminuir a probabilidade.

1.5 Principal Diferença entre o Uni e Multivariado

O GMM univariado modela apenas uma variável por vez, enquanto o GMM multivariado modela várias variáveis simultaneamente, capturando correlações entre elas.

1.6 Aplicações do GMM

1. Medicina:

É utilizado para auxiliar na identificação de padrões em conjuntos de dados médicos, e eles podem ser usados para detectar doenças, identificar grupos de pessoas com queixas comuns e até mesmo fazer previsões.

2. Análise do comportamento do consumidor:

Os GMMs podem ser usados em publicidade para realizar as análises de comportamento do usuário e com isso gerar previsões com base em dados anteriores sobre possíveis transações potenciais.

3. Previsão do preço das ações:

A previsão do preço das ações é mais uma aplicação dos modelos de mistura gaussiana, que podem ser utilizados em séries temporais de preços de ações na economia. Os modelos de mistura gaussiana podem ser empregados para identificar pontos de inflexão em análises de séries temporais e auxiliar na descoberta de momentos cruciais nos preços das ações ou outros movimentos de mercado.

4. Recursos audiovisuais:

Recentemente, os GMMs têm se mostrado eficazes na extração de características de dados de áudio para uso em sistemas de reconhecimento de fala. Eles também são importantes no rastreamento de múltiplos objetos, onde a quantidade de componentes na mistura e os valores de suas médias preveem a localização de um objeto em cada quadro de um vídeo, permitindo o rastreamento de objetos.

2. Demonstração da implementação algoritmo para alguma base de dados

Link do collab onde foi realizado a implementação do algoritmo: https://colab.research.google.com/drive/153_Y0wJnnyNNM87YUeU7OYYifxHNerDU?usp=sharing

```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.metrics import adjusted_rand_score, normalized_mutual_info_score
```

A base de dados escolhida para a implementação do algoritmo Gaussian Mixture Model (GMM) foi a base de dados da íris, onde, por ser famosa, já vem inclusa na biblioteca da Scikit-Learn, inclusive sendo bastante usada para o desenvolvimento do nosso algoritmo.

A Scikit-Learn possui tudo o que é necessário para a implementação do algoritmo GMM e posteriormente comparação com o kmeans.

```
iris = load_iris()
x = iris.data
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

## Ponto gerado para teste

novoPonto = [[5.9, 3, 5.1, 1.8]]
novoPontoScaled = scaler.transform(novoPonto)
```

Esse trecho do código visa carregar a base de dados da íris e realizar o pré-processamento dessa base de dados através da classe StandardScaler. Além disso, é criado um novo ponto, sendo um ponto fictício com dados semelhantes aos dados da íris, e consequentemente sendo pré-processado da mesma forma que os demais dados. Esse ponto possui o objetivo de ser um “teste” que mostrará a forma como o GMM decide qual dado será inserido em algum dos clusters.

```
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters_kmeans = kmeans.fit_predict(x_scaled)
```

Esse trecho do código é responsável por 2 etapas:

A primeira linha é criado um objeto que “configura” a forma como o kmeans vai ser usado, com 1° parâmetro indicando o número de clusters, o 2 parâmetro indicando uma “semente” (os centróides do kmeans vão iniciar no mesmo lugar), 3 parâmetro indicando o número de inicializações que o kmeans irá ser executado (número de vezes que o algoritmo vai ser executado).

A segunda linha representa os clusters e o agrupamento encontrados após a execução do kmeans (esse resultado vai ser armazenado em clusters_kmeans).

```
gmm = GaussianMixture(n_components=3, covariance_type='full', random_state=42, n_init=10)
clusters = gmm.fit_predict(x_scaled)
probPonto = gmm.predict_proba(novoPontoScaled)
```

Já nessa parte do código, é responsável por 3 etapas:

A primeira linha, assim como o kmeans, está sendo criado um objeto que configura a forma como o Gaussian Mixture Model vai ser usado para o dataset, os parâmetros fazem praticamente a mesma coisa que no caso do kmeans, com exceção de um novo parâmetro, o “ covariance_type=’full’ ”. Esse parâmetro é

responsável por informar à classe a forma como esse GMM irá configurar as elipses dos clusters, nesse caso, com o parâmetro “full”, cada elipse vai se comportar de forma diferente das outras, ou seja, cada cluster pode ter sua elipse se comportando de uma maneira diferente das outras, seja alongando, seja encolhendo, sendo rotacionado, entre outros.

A segunda linha é responsável por armazenar os clusters e o agrupamento encontrado a partir da execução do modelo GMM usando os dados já pre-processados.

A terceira linha é responsável por armazenar a probabilidade do ponto fictício criado anteriormente (o novoPonto), de pertencer à algum dos clusters.

O código posterior à esse ponto é a criação do gráfico para visualização dos clusters criados por ambos algoritmos.

3. Validação da implementação comparando a os resultado do k-means

Antes de efetuarmos um esboço gráfico da comparação entre o algoritmo GMM e o Kmeans, calculamos duas métricas que avaliam o quão semelhantes são as estruturas produzidas pelos algoritmos. O ARI(Adjusted Rand Index), mede o quão de acordos os dois modelos estão. ARI igual à 1 significa uma concordância perfeita. Já o NMI(Normalized Mutual) Information mede a quantidade de informação compartilhada entre os dois modelos. Valores próximos de 1 indicam um forte acordo.

```
# Métricas de validação

ari_score = adjusted_rand_score(clusters_kmeans, clusters)
nmi_score = normalized_mutual_info_score(clusters_kmeans, clusters)
print(f"ARI Score: {ari_score:.3f} | NMI Score: {nmi_score:.3f}\n")
```

Antes de iniciarmos o loop criamos a figura que definirá a área do desenho do gráfico. Logo após temos o loop que plota os pontos onde os modelos concordam. Na segunda linha é criado uma máscara lógica que seleciona apenas os pontos onde o KMeans e GMM atribuem o mesmo cluster em *i.* Quando há concordância, os pontos aparecem totalmente brancos com borda colorida

```
plt.figure(figsize=(10, 8))

for i in range(3):
    mask_both = (clusters_kmeans == i) & (clusters == i)
    plt.scatter(x_scaled[mask_both, 2], x_scaled[mask_both, 3],
                c='white', edgecolors=cores[i], s=120, linewidth=2.5,
                label=f'Acordo cluster {i}', alpha=0.9, zorder=3)
```

Agora é feito a comparação quando os modelos discordam, é criado uma máscara lógica para pontos onde o KMeans e GMM não concordam. Se existir, os pontos são representados em vermelho.

```
mask_discord = clusters_kmeans != clusters
plt.scatter(x_scaled[mask_discord, 2], x_scaled[mask_discord, 3],
            c='red', alpha=0.7, s=60, label='Discordância', zorder=2)
```

Nesta parte do código, plotamos o ponto fictício com uma visualização em formato de X para facilitar a identificação.

```
plt.scatter(novoPontoScaled[0, 2], novoPontoScaled[0, 3],
          s=200, c='black', marker='X', linewidth=3,
          label=f'0 - {probPonto[0][0]*100:.1f}%| 1 - {probPonto[0][1]*100:.1f}%| 2 - {probPonto[0][2]*100:.1f}', zorder=5)
```

E por último, temos os rótulos dos Eixos, o Título do gráfico, o posicionamento da legenda, as grades e o ajuste do layout. Por fim, temos a exibição final.

```
plt.xlabel("petal length (cm)")
plt.ylabel("petal width (cm)")
plt.title("Comparação KMeans vs GMM\n(Branco = Acordo | Vermelho = Discordância)", fontsize=14, pad=20)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

4. Análise dos resultados e conclusão geral

A análise dos resultados obtidos a partir da aplicação dos algoritmos **K-Means** e **Gaussian Mixture Model (GMM)** ao conjunto de dados Iris evidenciou

elevada similaridade entre as estruturas de clusterização geradas por ambos os métodos. Foi observado que o grupo correspondente à espécie Iris-setosa apresentou separação completa e bem definida nos dois modelos, indicando que suas características são linearmente separáveis das demais classes. Essa estabilidade estrutural demonstra que ambos os algoritmos são eficazes na identificação de padrões claramente distintos no conjunto de dados.

Entretanto, nas regiões correspondentes às espécies Iris-versicolor e Iris-virginica, verificou-se a presença de sobreposição entre os clusters, reflexo direto da proximidade estatística entre essas duas classes no espaço de características. Nessa região de fronteira, ocorreram as principais discordâncias entre os agrupamentos produzidos pelo K-Means e pelo GMM. Tal comportamento era esperado, uma vez que o K-Means realiza uma segmentação rígida baseada apenas na distância aos centróides, enquanto o GMM considera a distribuição probabilística dos dados, permitindo que um mesmo ponto possua graus distintos de pertencimento a diferentes clusters.

O gráfico de concordância e discordância reforça essa interpretação ao evidenciar que a maior parte dos pontos encontra-se em região de acordo entre os dois métodos, ao passo que os pontos de discordância concentram-se exclusivamente na zona de transição entre os clusters intermediário e superior. Essa concentração confirma que as divergências não representam falhas dos modelos, mas sim a incerteza natural presente nos dados nessa região do espaço amostral.

De forma geral, os resultados demonstram que o **K-Means e o GMM apresentam desempenho semelhante** na organização global dos dados, com alta concordância estrutural.

O GMM se mostra superior do ponto de vista estatístico e interpretativo, por ser capaz de representar a incerteza inerente aos dados e produzir classificações probabilísticas, especialmente em regiões onde há sobreposição entre os grupos. Assim, conclui-se que, embora ambos os métodos sejam adequados para a clusterização do conjunto Iris, o **GMM** oferece uma **modelagem mais rica e informativa em cenários de maior complexidade estatística.**

Referências

CASTRO, Leandro Nunes de; FERRARI, Daniel Gomes. Introdução à mineração de dados:
conceitos básicos, algoritmos e aplicações. São Paulo: Saraiva, v. 5, p. 1-376, 2016.
Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Machine Learning: Fundamental Concepts and Algorithms, 2nd Edition, Cambridge University Press, March 2020. ISBN: 978-1108473989.

https://dataminingbook.info/
https://nixustechnologies.com/gaussian-mixture-model/
https://towardsdatascience.com/gaussian-mixture-models-gmms-from-theory-to-implementation-4406c7fe9847/
https://www.ibm.com/think/topics/gaussian-mixture-model
https://towards datascience.com/gaussian-mixture-models-gmms-from-theory-to-implementation-4406c7fe9847/

Link do Slide:

https://docs.google.com/presentation/d/1JconN6mmfv_A97qctUfQifX2rPPQAp-luiydZBUIXrU/edit?usp=sharing