Aprendizado de Máquina – IMD1101

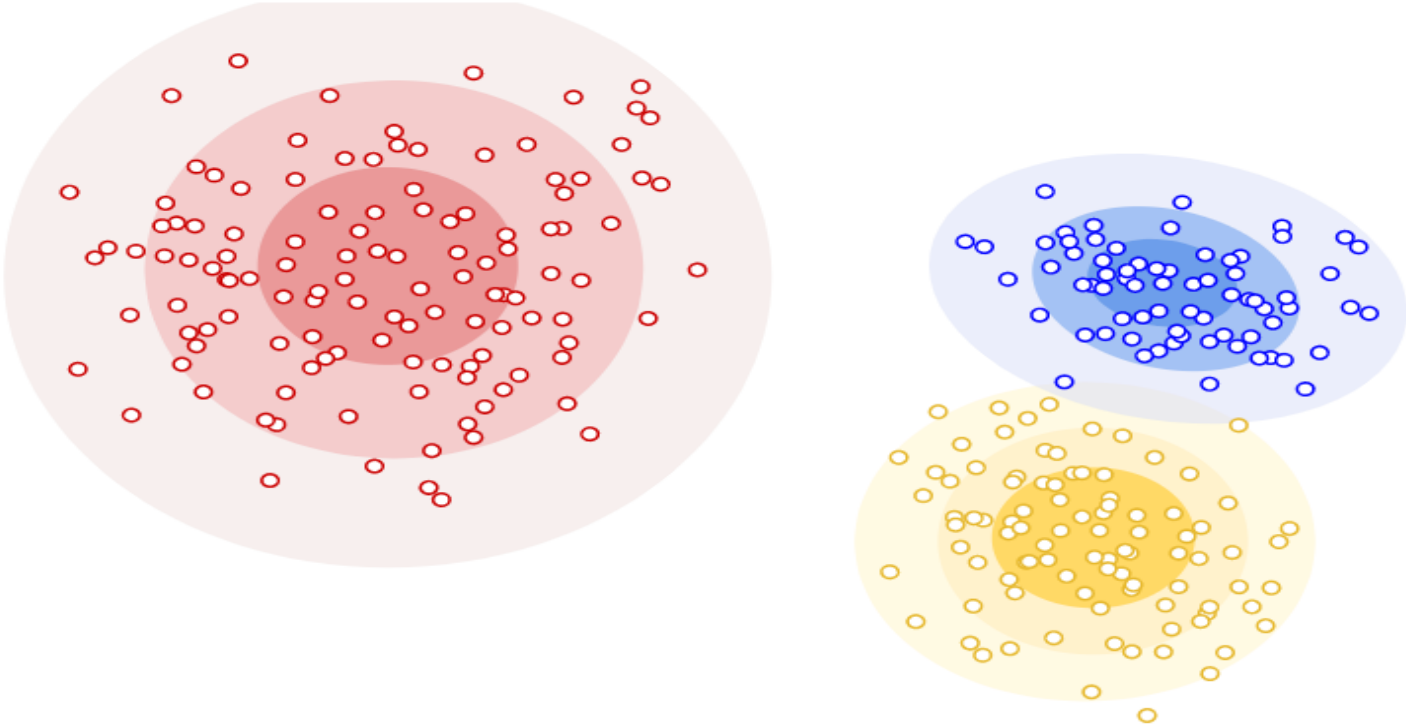
Aula 23 – Aprendizado Não Supervisionado 03

INSTITUTO METRÓPOLE DIGITAL

DIMAp UFRN

Algoritmos de Clustering

- Distribution-based Clustering:



https://developers.google.com/machine-learning/clustering/clustering-algorithms

Expectation Maximization
EM

Modelo de Misturas

- Resulta na **distribuição de probabilidade** de um ponto p(x):

$$p(x) = \sum_{k=1}^{K} P(Z_k)\, p(x|Z_k)$$

onde $\boldsymbol{p(x|Z_k)}$ é qualquer distribuição (**gaussiana**, poisson, exponencial, etc).

Modelo de Misturas

- O modelo de mistura mais comum é o de **gaussianas** – Gaussian mixture models (GMM).
- Modelo do cluster (grupo): distribuição normal (média μ e variância σ^2).
- Algoritmo EM (Expectation Maximization) é capaz de encontrar um **ótimo local** através da função de máxima verossimilhança de uma mistura de Gaussianas.

Expectation Maximization (EM)

- Gera-se (μ_a, σ_a^2) e (μ_b, σ_b^2) para k = 2;
- Para cada ponto de dado, $P(b|x_i)$? ou $P(a|x_i)$?
- Ajusta-se (μ_a, σ_a^2) e (μ_b, σ_b^2) para aproximar os pontos atribuídos às gaussianas.

https://www.youtube.com/watch?v=REypj2sy_5U

Algoritmo EM

- **Expectation Step**:
  - Nessa etapa, ele calcula a probabilidade de **cada ponto** pertencer a cada **Gaussiana**.
  - Além disso, é calculada uma nova estimativa da função de verossimilhança utilizando a equação:

$$P(d_i \in c_k) = w_k \Pr(d_i \mid c_k) \Big/ \sum_j w_j \Pr(d_i \mid c_j)$$

$$w_k = \frac{\sum_i \Pr(d_i \in c_k)}{N}$$

Algoritmo EM

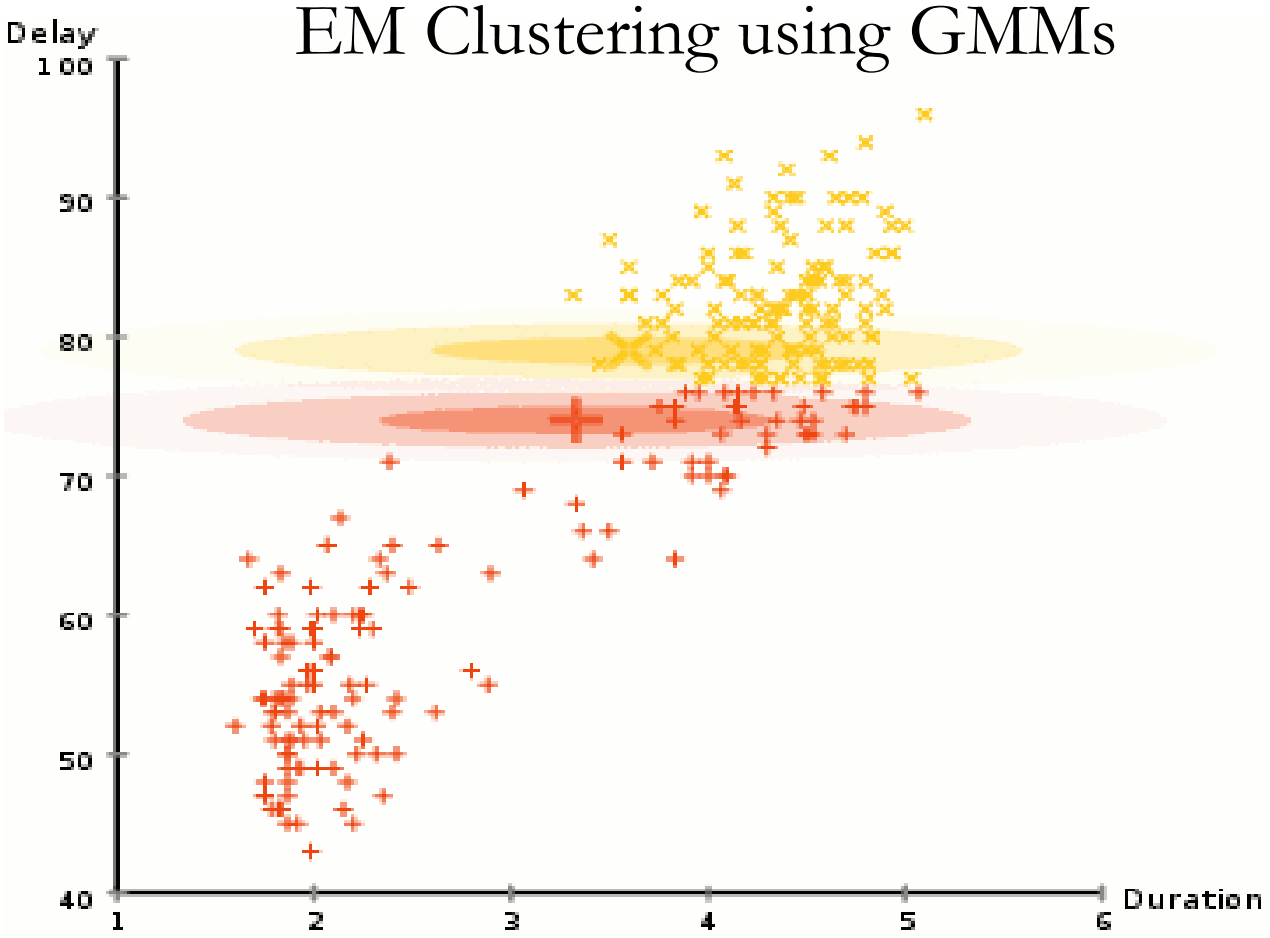
- **Maximation step**:
  - Nessa etapa, as componentes da mistura são maximizadas (estimativa dos parâmetros do modelo).

$$\mu_k = \frac{1}{m} \sum_{i=1}^{m} \frac{d_i P(d_i \in c_k)}{\sum_k P(d_i \in c_j)}$$

EM: passo a passo

1. Escolha um **valor para *k*** (grupos) e inicialize aleatoriamente os parâmetros da distribuição Gaussiana para cada grupo.
2. A partir das distribuições gaussianas para cada cluster, calcule a probabilidade de cada ponto de dados pertencer a um cluster específico (**Expectation Step**).
3. Com base nessas probabilidades, calculamos um novo conjunto de parâmetros para as distribuições gaussianas, de modo que maximizemos as probabilidades dos pontos de dados nos clusters (**Maximation step**).
4. As etapas 2 e 3 são repetidas até a convergência, em que as distribuições não mudam muito de iteração para iteração.

Algoritmo EM



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Considerações sobre EM

- Parâmetro ***k*** é definido a priori.
  - Por que?
- Converge para um **mínimo local**.
  - Função de máxima verossimilhança.

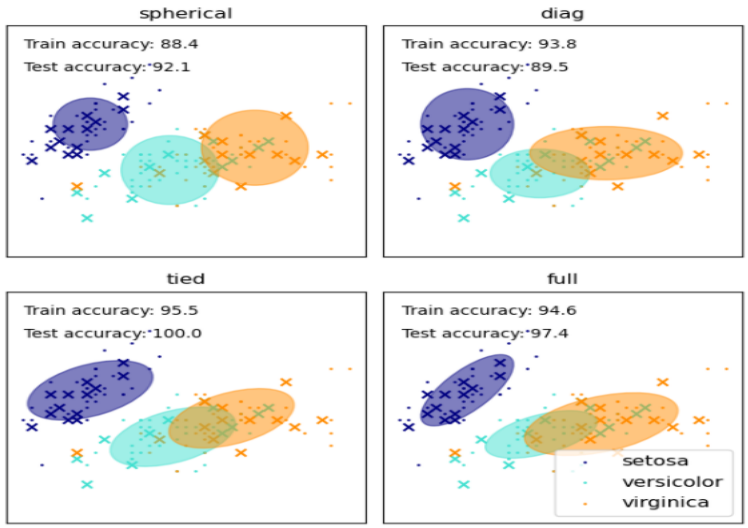
EM

- Utilizando **EM** (**scikit-learn**):

2.1.1. Gaussian Mixture

The `GaussianMixture` object implements the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A `GaussianMixture.fit` method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it mostly probably belong to using the `GaussianMixture.predict` method.

The `GaussianMixture` comes with different options to constrain the covariance of the difference classes estimated: spherical, diagonal, tied or full covariance.



https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

EM

- Utilizando o **GaussianMixture**:

```
import pandas as pd
import numpy as np
from google.colab import files
import io
from sklearn.mixture import GaussianMixture #EM

uploaded = files.upload()
dados = pd.read_csv(io.BytesIO(uploaded['PessoaNormBinary.csv']))

# ## Expectation Maximization
gmm = GaussianMixture(n_components=2, covariance_type='full')
gmm.fit(dados)
gmm.fit_predict(dados)

# pegando os labels
dados["Cluster"] = gmm.predict(dados)

# juntando os labels com o restante do dataset
dados["Cluster"] = 'cluster' + dados["Cluster"].astype(str)

# Salvando Pessoa.csv transformado
df = pd.DataFrame(dados)
df.to_csv('PessoaNorm_EM_k2.csv')

# Download do arquivo transformado
files.download('PessoaNorm_EM_k2.csv')
```

EM01.py

Análise de Cluster

Analisando as Partições

Escolher arquivos PessoaNormBinary.csv

- **PessoaNormBinary.csv**(text/csv) - 625 bytes, last modified: 09/03/2019 - 100% done

```
Saving PessoaNormBinary.csv to PessoaNormBinary (11).csv
DB - Hierarquico_c  (k = 2): 1.821
DB - Hierarquico_c  (k = 3): 1.311
DB - Hierarquico_s (k = 2): 0.679
DB - Hierarquico_s (k = 3): 0.813
DB - Hierarquico_av (k = 2): 0.679
DB - Hierarquico_av (k = 3): 1.099
DB - k-means_s10 (k = 2): 1.467
DB - k-means_s10 (k = 3): 1.311
DB - k-means_s37 (k = 2): 1.467
DB - k-means_s37 (k = 3): 1.316
DB - EM (k = 2): 1.832
DB - EM (k = 3): 1.311
```

Partição	Grupos	#	DB	Partição	Grupos	#	DB
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2	1	0,6790	PessoaNormBinary_kM-s10-2k	2	6	1,4670
	1	6			1	4	
	2	1			2	5	
PessoaNormBinary_Ha-avLink-3k	3	3	1,0990	PessoaNormBinary_kM-s10-3k	3	1	1,3110
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	1,8210	PessoaNormBinary_kM-s37-2k	2	5	1,4670
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	1,3110	PessoaNormBinary_kM-s37-3k	3	4	1,3160
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2	1	0,6790	PessoaNormBinary_EM-2k	2	5	1,8320
	1	8			1	4	
	2	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3	1	0,8130	PessoaNormBinary_EM-3k	3	3	1,3160

Analisando as Partições

Partição	Grupos	#	DB	Partição	Grupos	#	DB
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2	1	0,6790	PessoaNormBinary_kM-s10-2k	2	6	1,4670
	1	6			1	4	
	2	1			2	5	
PessoaNormBinary_Ha-avLink-3k	3	3	1,0990	PessoaNormBinary_kM-s10-3k	3	1	1,3110
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	1,8210	PessoaNormBinary_kM-s37-2k	2	5	1,4670
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	1,3110	PessoaNormBinary_kM-s37-3k	3	4	1,3160
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2	1	0,6790	PessoaNormBinary_EM-2k	2	5	1,8320
	1	8			1	4	
	2	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3	1	0,8130	PessoaNormBinary_EM-3k	3	3	1,3160

Resultado Final

Partição	Grupos	#	DB	Partição	Grupos	#	DB
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2	1	0,6790	PessoaNormBinary_kM-s10-2k	2	6	1,4670
	1	6			1	4	
	2	1			2	5	
PessoaNormBinary_Ha-avLink-3k	3	3	1,0990	PessoaNormBinary_kM-s10-3k	3	1	1,3110
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	1,8210	PessoaNormBinary_kM-s37-2k	2	5	1,4670
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	1,3110	PessoaNormBinary_kM-s37-3k	3	4	1,3160
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2	1	0,6790	PessoaNormBinary_EM-2k	2	5	1,8320
	1	8			1	4	
	2	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3	1	0,8130	PessoaNormBinary_EM-3k	3	3	1,3160

Analisando as Partições

- **PessoaNormBinary.csv**(text/csv) - 625 bytes, last modified: 09/03/2019 - 100% done

```
Saving PessoaNormBinary.csv to PessoaNormBinary (2).csv
Hierarquico Complete 2k - Silhouette=> 0.110
Hierarquico Complete 3k - Silhouette=> 0.181
Hierarquico Single 2k - Silhouette=> 0.143
Hierarquico Single 3k - Silhouette=> -0.003
Hierarquico Average 2k - Silhouette=> 0.143
Hierarquico Average 3k - Silhouette=> 0.158
k-Means_s10 2k - Silhouette=> 0.146
k-Means_s10 3k - Silhouette=> 0.181
k-Means_s37 2k - Silhouette=> 0.146
k-Means_s37 3k - Silhouette=> 0.181
EM 2k - Silhouette=> 0.114
EM 3k - Silhouette=> 0.140
```

Partição	Grupos	#	SG	Partição	Grupos	#	SG
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2	1	0,1430	PessoaNormBinary_kM-s10-2k	2	6	0,1460
	1	6			1	4	
	2	3			2	5	
PessoaNormBinary_Ha-avLink-3k	3	1	0,1580	PessoaNormBinary_kM-s10-3k	3	1	0,1810
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	0,1100	PessoaNormBinary_kM-s37-2k	2	5	0,1460
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	0,1810	PessoaNormBinary_kM-s37-3k	3	4	0,1810
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2	1	0,1430	PessoaNormBinary_EM-2k	2	5	0,1140
	1	8			1	4	
	2	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3	1	-0,0030	PessoaNormBinary_EM-3k	3	3	0,1400

Analisando as Partições

Partição	Grupos	#	SG	Partição	Grupos	#	SG
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2 →	1	0,1430	PessoaNormBinary_kM-s10-2k	2	6	0,1460
	1	6			1	4	
	2	3			2	5	
PessoaNormBinary_Ha-avLink-3k	3 →	1	0,1580	PessoaNormBinary_kM-s10-3k	3 →	1	0,1810
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	0,1100	PessoaNormBinary_kM-s37-2k	2	5	0,1460
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	0,1810	PessoaNormBinary_kM-s37-3k	3	4	0,1810
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2 →	1	0,1430	PessoaNormBinary_EM-2k	2	5	0,1140
	1	8			1	4	
	2 →	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3 →	1	-0,0030	PessoaNormBinary_EM-3k	3	3	0,1400

Resultado Final

Partição	Grupos	#	SG	Partição	Grupos	#	SG
	1	9			1	4	
PessoaNormBinary_Ha-avLink-2k	2	1	0,1430	PessoaNormBinary_kM-s10-2k	2	6	0,1460
	1	6			1	4	
	2	3			2	5	
PessoaNormBinary_Ha-avLink-3k	3	1	0,1580	PessoaNormBinary_kM-s10-3k	3	1	0,1810
	1	4			1	5	
PessoaNormBinary_Ha-cLink-2k	2	6	0,1100	PessoaNormBinary_kM-s37-2k	2	5	0,1460
	1	4			1	4	
	2	3			2	2	
PessoaNormBinary_Ha-cLink-3k	3	3	0,1810	PessoaNormBinary_kM-s37-3k	3	4	0,1810
	1	9			1	5	
PessoaNormBinary_Ha-sLink-2k	2	1	0,1430	PessoaNormBinary_EM-2k	2	5	0,1140
	1	8			1	4	
	2	1			2	3	
PessoaNormBinary_Ha-sLink-3k	3	1	-0,0030	PessoaNormBinary_EM-3k	3	3	0,1400

Obrigado!!!