**I. Pen-and-paper**

1. Vamos calcular uma época do algoritmo Expectation Maximization Clustering, de forma a atualizar os clusters fornecidos no enunciado, para que melhor classifiquem o dataset. Para isso vamos aplicar os seguintes dois passos:

1º - Atribuir classes aos pontos

Usando o teorema de Bayse vamos calcular a probabilidade de cada ponto pertencer a um dos clusters

Antes de calcular o posteior temos de determinar as seguintes parcelas:

- Likelihoods = d

- Joint Probability =

Todos os parâmetros necessários, são fornecidos no enunciado falta então apenas calcular os posteriors para cada instância do dataset, estas probabilidades é que vão definir a qual cluster a instância pertence

-

*Cluster*

- **Likelihood**:

- **Joint Probability**:

*Cluster*

- **Likelihood**:

- **Joint Probability**:

- **Denominador**:

- **Posterior**: - **Posterior**:

-

*Cluster*

- **Likelihood**:

- **Joint Probability**:

*Cluster*

- **Likelihood**:

- **Joint Probability**:

- **Denominador**:

- **Posterior**: - **Posterior**:

-

*Cluster*

- **Likelihood**:

- **Joint Probability**:

*Cluster*

- **Likelihood**:

- **Joint Probability**:

- **Denominador**:

- **Posterior**: - **Posterior**:

-

*Cluster*

- **Likelihood**:

- **Joint Probability**:

*Cluster*

- **Likelihood**:

- **Joint Probability**:

- **Denominador**:

- **Posterior**: - **Posterior**:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Analisando o posterior calculado, podemos verificar que as instâncias do dataset pertencem aos seguintes clusters:

2º - Maximization step (atualizar os parâmetros que definem o cluster de forma a melhor separar o dataset)

De forma a tornar os cálculos mais simples, vamos organizar os posteiros em dois vetores (um para cada cluster)

Estimar Priors

Estimar

Estimar

A picture containing diagram

Description automatically generated

Esta imagem mostra a forma dos novos clusters calculados usando o método EM

1. Usando as classificações dos pontos calculadas no exercício anterior podemos então calcular a silhueta de cada cluster e consequentemente a geral

Onde:

Distância média entre x e os pontos que pertencem ao mesmo cluster

Distância média entre x e os pontos do cluster mais próximo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 4.038843615 | 0 (único ponto do cluster) | 4.495358041 | 4.928650381 |
|  | 8.544003745 | 6.982375994 | 6 | 6.403124237 |
|  | 0.52728911 | 1 | 0.25077366 | 0.230274129 |

A silhueta de um cluster e dada pela média aritmética das silhuetas dos pontos que lhe pertencem, assim:

Da mesma forma, a silhueta do conjunto de clusters é a média aritmética da silhueta de cada cluster

1. De forma simplificar a resolução deste problema vamos primeiro aproximar a VC-dimension para um espaço geral com N features. Esta aproximação vai ser feita usando o número de parâmetros necessários para cada classificador, já que:

i.) MLP com 3 hidden layers onde cada camada possui N (número de features) nodes

1. Temos uma matriz , parâmetros, e um vetor parâmetros
2. Temos uma matriz , parâmetros, e um vetor parâmetros
3. Temos uma matriz , parâmetros, e um vetor parâmetros
4. Temos uma matriz parâmetros, e um vetor parâmetros

Assim, a classificação usando uma MLP necessita de parâmetros que aproxima o valor da VC-dimension

ii) Árvore de Decisão onde as variáveis são discreteadas em 3 bins

Assim o número total de combinações possíveis (3 possibilidades para cada feature). Para classificar estas instâncias basta contruir onde cada folha contém a classificação para uma destas instâncias. Devido a natureza da árvore de decisão conseguimos classificar perfeitamente qualquer um destes pontos (cada

um vai corresponder a um ramo da árvore), assim podemos concluir que a VC-dimension da árvore de decisão é igual a .

iii) Baysian classifier com likelihoods com distribuição normal multivariada

Como:

Assim os parametros das likelihoods são os mesmo da normal multivariada:

- = vetor de médias (), corresponde então a N parâmetros distintos

- = matriz de covariância (), mas a matriz é simétrica logo apenas corresponde a parâmetros distintos

Assim uma classificação Baysiana necessita de parâmetros que a próxima o valor da VC-dimension

Tendo então obtido uma fórmula para a VC-dimension em função no número de features torna a resolução dos seguintes exercícios trivial

1. Para N = 5, temos que
2. VC-dimension = 102
3. VC-dimension = 243
4. VC-dimension = 41
5. Chart, line chart

   Description automatically generatedChart, line chart

   Description automatically generatedPodemos verificar que a VC-dimension da árvore de decisão é exponencialmente maior que os restantes dois classificadores
6. O grau de crescimento dos dois classificadores é semelhante (crescimento quadrático

**II. Programming and critical analysis**

* 1. Aplicando o teste ECR ao clustering realizado por k-means, obtém-se que para k=2 o valor de ECR é 13,5, e para k=3, o valor de ECR é 6,6(6). Comparando estes resultados é notável que quando são feitos 3 clusters, o agrupamento de dados está menos sujeito a agrupamentos incorretos, pois é o que obtém menor valor de ECR.
  2. Para k = 2, o silhouette coefficient é 0.5967981179111456, e, para k = 3, o silhouette coeffienct é 0.5256774849851862. Estes resultados são bastante similares, mas k = 2 tem um valor ligeiramente maior que o de k = 3, o que leva a concluir que para k = 2, os clusters estão mais bem definidos, ainda que não seja algo significativo.

Chart, scatter chart

Description automatically generatedz

1. A partir da imagem acima, é possível observar que os clusters não se encontram devidamente distanciados, nem bem-formados, pelo que as fronteiras entre os clusters não se conseguem trivialmente delinear. Isto afeta a coesão dos conjuntos e tal pode levar a que não sejam tomadas decisões corretas.

**III. APPENDIX**

def ECR(label, outputs, k):

    benign = [0, 0, 0]

    malign = [0, 0, 0]

    aux = 0

    for cluster in range(0, k):

        for i in range(0, len(label)):

            if label[i] == cluster:

                if outputs[i] == "benign":

                    benign[cluster]+=1

                else:

                    malign[cluster]+=1

    for cluster in range(0, k):

        max\_val = max(benign[cluster], malign[cluster])

        aux += benign[cluster] + malign[cluster] - max\_val

    return aux/k

data = arff.loadarff("C:\\Users\\print\\Downloads\\breast.w.arff")

dataset = pandas.DataFrame(data[0])

dataset["Class"] = dataset["Class"].str.decode('utf-8')

inputs = dataset.drop(columns=["Class"]).values

outputs = dataset["Class"].values

kMeans2 = KMeans(n\_clusters = 2, random\_state = 13)

kMeans3 = KMeans(n\_clusters = 3, random\_state = 13)

kMeans2.fit(inputs)

kMeans3.fit(inputs)

label2 = kMeans2.predict(inputs)

label3 = kMeans3.predict(inputs)

print(f"ECR score n=2: {ECR(label2, outputs, 2)}")

print(f"ECR score n=3: {ECR(label3, outputs, 3)}")

print(f"Silhouette score n=2: {silhouette\_score(inputs, label2)}")

print(f"Silhouette score n=3: {silhouette\_score(inputs, label3)}")

kbest = SelectKBest(feature\_selection.mutual\_info\_classif, k = 2)

kbest.fit(inputs, outputs)

inputsTopFeatures = inputs[:, kbest.get\_support(indices = True)]

newKMeans3 = KMeans(n\_clusters = 3, random\_state = 13).fit(inputs)

newKMeans3\_1 = KMeans(n\_clusters = 3, random\_state = 13).fit(inputsTopFeatures)

labelScatter = newKMeans3.predict(inputs)

centroids = newKMeans3\_1.cluster\_centers\_

plt.scatter(inputsTopFeatures[:,0], inputsTopFeatures[:,1], c = newKMeans3.labels\_, cmap = 'rainbow', alpha = 0.20)

plt.scatter(centroids[2][0], centroids[2][1], marker = "x", label = "Centroid of cluster 0", s = 150, linewidths= 2, zorder = 10, c = "cyan")

plt.scatter(centroids[0][0], centroids[0][1], marker = "x", label = "Centroid of cluster 1", s = 150, linewidths= 2, zorder = 10, c = "violet")

plt.scatter(centroids[1][0], centroids[1][1], marker = "x", label = "Centroid of cluster 2", s = 150, linewidths= 2, zorder = 10, c = "red")

plt.legend()

plt.show()

**END**