**I. Pen-and-paper**

1. Cálculo de probabilidades e probabilidades conjuntas

=

=

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

Exemplificação de cálculos para a primeira linha

Normalização de *posterioris*

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  | 0.99999 |
|  | 0.98271 | 0.01729 |
|  | 0.85698 | 0.14302 |

Exemplificação de cálculos para a primeira linha

Cálculo dos parâmetros para os novos clusters

0.7099

1. De acordo com os valores obtidos na normalização de *posterioris* podemos concluir que os pontos e pertencem aos *cluster* 1 enquanto que o ponto pertenceao *cluster* 2 devido à maior probabilidade observada para estes *clusters*.

Dado que a o valor da silhueta pode variar entre -1 e 1 e tendo em conta o valor obtido, podemos afirmar que o nível de silhueta não é muito elevado o que significa que os clusters obtidos não se encontram perfeitamente separados e coesos.

1. a)
2. De forma a identificar a dimensão-VC do classificador apresentado iremos identificar o número de parâmetros necessários uma vez que é um bom estimador da dimensão-VC do classificador em questão.

No caso do MLP com 3 camadas internas com tantos nós quantas variáveis de input (5 variáveis neste caso), obtemos que serão necessárias 4 matrizes de pesos e 4 *bias*.

Devido à arquitetura do MLP, quer a matriz de pesos entre a camada de input e a primeira camada interna, quer as matrizes de pesos entre camadas internas serão compostas por 5\*5 parâmetros. Já os *bias* para as 3 camadas internas são compostos por 5 elementos.

A matriz de pesos entre a última camada interna e a camada final é composta por 2\*5 elementos e o *bias* da camada de output é composto por 2 elementos.

Assim, o número de parâmetros do classificador é dado por .

Logo a dimensão-VC estimada é de 102.

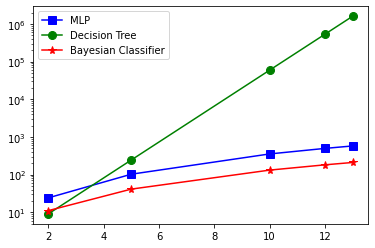
1. Para uma árvore de decisão com 5 variáveis discretizadas em 3 *bins*, podemos facilmente notar que no máximo a árvore pode conter nós, conseguindo separar, no máximo, observações distintas. Logo, a dimensão-VC é de , .
2. Para o classificador Bayesiano iremos novamente utilizar número de parâmetros para estimar a dimensão-VC.

Para o treino do classificador em questão necessitamos de um dos *priors* (uma vez que o outro pode ser calculado a partir do obtido), de um vetor de 5 elementos com a média para cada uma das 2 classes de output e de uma matriz de covariâncias para cada classe com 5\*5 elementos cada. No entanto, devido à simetria da matriz de covariâncias são necessários apenas parâmetros para a sua construção completa.

No total, são necessários parâmteros, logo a dimensão-VC aproxima-se de 41

|  |  |  |  |
| --- | --- | --- | --- |
| Dimensionalidade | MLP | Decision Tree | Bayesian Classifier |
| 2 |  |  |  |
| 5 |  | 243 | 41 |
| 10 | 352 | 59049 | 131 |
| 12 | 494 | 531441 | 181 |
| 13 | 574 | 159423 | 209 |

b) Fórmulas utilizadas

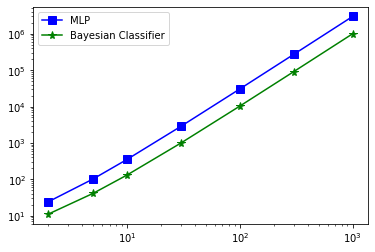


Ao observar o gráfico ao lado podemos observar que a dimensão-VC da árvore de decisão cresce muito mais rapidamente com o aumento da dimensionalidade dos dados que os outros dois classificadores (tal como as fórmulas acima sugeriam).

|  |  |  |
| --- | --- | --- |
| Dimensionalidade | MLP | Bayesian Classifier |
| 2 | 24 | 11 |
| 5 | 102 | 41 |
| 10 | 352 | 131 |
| 30 | 2852 | 991 |
| 100 | 30502 | 10301 |
| 300 | 271502 | 90901 |
| 1000 | 3005002 | 1003001 |

c)

Fórmulas utilizadas

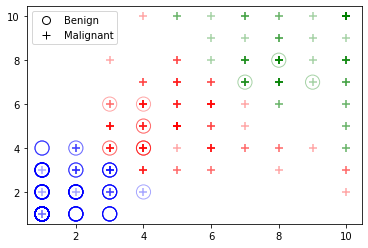


Ao observar o gráfico ao lado podemos observar que a dimensão-VC do MLP é superior à dimensão-VC do classificador Bayesiano apesar de ambos os modelos terem um crescimento semelhante com o aumento da dimensionalidade.

1. a) Ao comparar as soluções produzidas para 2 e 3 *clusters* tendo em conta a medida ECR, verificamos que a solução com 3 *clusters* tem um valor menor (6,666(6)) do que a solução com apenas 2 *clusters* (13.5). Pelo ECR, dado que se trata de uma medida externa podemos concluir que a solução com 3 *clusters* separa melhor as classes das observações.

b) Ao comparar as soluções produzidas para 2 e 3 *clusters* tendo em conta a sua silhueta, verificamos que a solução com 2 *clusters* tem um valor mais próximo de 1 (0.596798) do que a solução com 3 clusters (0.52495). Pela silhueta, dado que se trata de uma medida interna que varia entre -1 e 1 podemos concluir que a solução com 2 clusters forma *clusters* mais bem separados e mais coesos.

1. Representação gráfica da distribuição dos 3 clusters (cada cor corresponde a um cluster) e das classes das observações:

****

1. Ao observar o gráfico produzido no exercício anterior, podemos observar que os *clusters* formados são bastante coesos uma vez que as observações que lhes pertencem se encontram próximas entre si. No entanto, os mesmos *clusters* não se encontram bem separados uma vez que existem várias observações perto das fronteiras entre eles.

Analisando a distribuição de classes pelos 3 *clusters*, podemos concluir ainda que o *cluster* azul contém maioritariamente observações classificadas como benignas ao contrário dos *clusters* vermelho e verde que contém na sua grande maioria observações classificadas com malignas.

Assim, após esta análise, podemos afirmar que, apesar de os *clusters* formados não se encontrarem bem separados, a seleção de variáveis efetuada resultou numa solução de boa qualidade.

**III. APPENDIX**

**import** pandas **as** pd

**from** scipy.io **import** arff

**from** sklearn **import** cluster

**from** sklearn.metrics **import** silhouette\_score

**import** numpy **as** np

data = arff.loadarff('breast.w.arff')

df = pd.DataFrame(data[0])

df.dropna(inplace=True)

df.replace(b'benign', 0, inplace=True)

df.replace(b'malignant', 1, inplace=True)

data = df.drop(["Class"],axis=1).values

target = df["Class"].values

**def** ECR(labels, target, k):

c\_counts = []

**for** i **in** **range**(k):

ci\_filter = []

**for** j **in** **range**(**len**(labels)):

ci\_filter.append(labels[j]==i)

c\_counts.append(**min**(np.bincount(target[ci\_filter])))

error = 0

**for** el **in** c\_counts:

error += el

**return** error/k

**for** k **in** (2,3):

cl = cluster.KMeans(n\_clusters=k)

predictions = cl.fit\_predict(data)

**print**(k, "Clusters")

**print**(" ECR ", ECR(predictions,target,k))

**print**(" Silhouette ", silhouette\_score(data,predictions))

**from** sklearn.feature\_selection **import** SelectKBest, mutual\_info\_classif

**import** matplotlib.pyplot **as** plt

**import** matplotlib.lines **as** mlines

data\_selected = SelectKBest(mutual\_info\_classif, k=2).fit\_transform(data, target)

cl = cluster.KMeans(n\_clusters=3)

pred = cl.fit\_predict(data\_selected)

clusters = [[[],[]],[[],[]]], [[[],[]],[[],[]]], [[[],[]],[[],[]]]

**for** i **in** **range**(**len**(data\_selected)):

clusters[pred[i]][target[i]][0].append(data\_selected[i].item(0))

clusters[pred[i]][target[i]][1].append(data\_selected[i].item(1))

plt.scatter(clusters[0][0][0],clusters[0][0][1],marker="o",s=200,alpha=0.35,color="none",edgecolors="red")

plt.scatter(clusters[0][1][0],clusters[0][1][1],marker="+",s=60 ,alpha=0.35,color="red")

plt.scatter(clusters[1][0][0],clusters[1][0][1],marker="o",s=200,alpha=0.35,color="none",edgecolors="blue")

plt.scatter(clusters[1][1][0],clusters[1][1][1],marker="+",s=60 ,alpha=0.35,color="blue")

plt.scatter(clusters[2][0][0],clusters[2][0][1],marker="o",s=200,alpha=0.35,color="none",edgecolors="green")

plt.scatter(clusters[2][1][0],clusters[2][1][1],marker="+",s=60 ,alpha=0.35,color="green")

leg1 = mlines.Line2D([], [], marker='o',color="none",markeredgecolor="black",

markersize=8, label='Benign')

leg2 = mlines.Line2D([], [], marker='+',color="none",markeredgecolor="black",

markersize=8, label='Malignant')

plt.legend(handles=[leg1,leg2])

**END**