TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 053

I. Pen-and-paper

1) Each EM-clustering iteration has two steps. Considering only one iteration we have:

E-Step: Assign each point to cluster that yields higher posterior.

- For x_1:

- For cluster c_1:
  * Prior: $p(c_1 = 1) = \pi_1 = 0.7$
  * Likelihood: $p(x_1|c_1 = 1) = \frac{1}{2\pi}\frac{1}{\sqrt{det(\Sigma_1)}} exp\left(-\frac{1}{2}(x_1 - \mu_1)^T(\Sigma_1{}^{-1})(x_1 - \mu_1)\right) =$ $\frac{1}{2\pi}\frac{1}{1} exp\left(-\frac{1}{2}\left(\binom{2}{4} - \binom{2}{4}\right)^T \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\left(\binom{2}{4} - \binom{2}{4}\right)\right) = \frac{1}{2\pi}$
  * Joint Probability: $p(c_1 = 1, x_1) = p(c_1 = 1)p(x_1|c_1 = 1) = 0.7 \times \frac{1}{2\pi} = 0.11140846016$

- For cluster c_2:
  * Prior: $p(c_2 = 1) = \pi_2 = 0.3$
  * Likelihood: $p(x_1|c_2 = 1) = \frac{1}{2\pi}\frac{1}{\sqrt{det(\Sigma_2)}} exp\left(-\frac{1}{2}(x_1 - \mu_2)^T(\Sigma_2{}^{-1})(x_1 - \mu_2)\right) =$ $\frac{1}{2\pi}\frac{1}{2} exp\left(-\frac{1}{2}\left(\binom{2}{4} - \binom{-1}{-4}\right)^T \begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\left(\binom{2}{4} - \binom{-1}{-4}\right)\right) = 9.43877951346626 \times 10^{-10}$
  * Joint Probability: $p(c_2 = 1, x_1) = p(c_2 = 1)p(x_1|c_2 = 1) = 0.3 \times 9.43877951346626 \times 10^{-10} = 2.831633854039878 \times 10^{-10}$

- So, we can compute the normalized posteriors for each cluster:

$$c_1: p(c_1 = 1|x_1) = \frac{p(c_1 = 1, x_1)}{p(c_1 = 1, x_1) + p(c_2 = 1, x_1)} = 0.9999999974583315$$
$$c_2: p(c_2 = 1|x_1) = \frac{p(c_2 = 1, x_1)}{p(c_1 = 1, x_1) + p(c_2 = 1, x_1)} = 2.541668597399302 \times 10^{-9}$$

- For x_2:

- For cluster c_1:
  * Prior: $p(c_1 = 1) = \pi_1 = 0.7$
  * Likelihood: $p(x_2|c_1 = 1) = \frac{1}{2\pi}\frac{1}{\sqrt{det(\Sigma_1)}} exp\left(-\frac{1}{2}(x_2 - \mu_1)^T(\Sigma_1{}^{-1})(x_2 - \mu_1)\right) = 2.2390899578253236 \times 10^{-17}$
  * Joint Probability: $p(c_1 = 1, x_2) = p(c_1 = 1)p(x_2|c_1 = 1) = 1.5673629704777265 \times 10^{-17}$

- For cluster c_2:
  * Prior: $p(c_2 = 1) = \pi_2 = 0.3$
  * Likelihood: $p(x_2|c_2 = 1) = 0.07957747154594767$
  * Joint Probability: $p(c_2 = 1, x_2) = p(c_2 = 1)p(x_2|c_2 = 1) = 0.023873241463784303$

- So, we can compute the normalized posteriors for each cluster:

$$c_1: p(c_1 = 1|x_2) = \frac{p(c_1 = 1, x_2)}{p(c_1 = 1, x_2) + p(c_2 = 1, x_2)} = 6.565354658081997 \times 10^{-16}$$

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 053

$$c_2: p(c_2 = 1|x_2) = \frac{p(c_2 = 1, x_2)}{p(c_1 = 1, x_2) + p(c_2 = 1, x_2)} = 0.9999999999999992$$

- For x_3:

- For cluster c_1:
  * Prior: $p(c_1 = 1) = \pi_1 = 0.7$
  * Likelihood: $p(x_3|c_1 = 1) = \frac{1}{2\pi}\frac{1}{\sqrt{det(\Sigma_1)}} exp\left(-\frac{1}{2}(x_3 - \mu_1)^T(\Sigma_1{}^{-1})(x_3 - \mu_1)\right) = 0.00023927977920047084$
  * Joint Probability: $p(c_1 = 1, x_3) = p(c_1 = 1)p(x_3|c_1 = 1) = 0.00016749584544032957$
- For cluster c_2:
  * Prior: $p(c_2 = 1) = \pi_2 = 0.3$
  * Likelihood: $p(x_3|c_2 = 1) = 9.82064017319871 \times 10^{-6}$
  * Joint Probability: $p(c_2 = 1, x_3) = p(c_2 = 1)p(x_3|c_2 = 1) = 2.946192051959613 \times 10^{-6}$
- So, we can compute the normalized posteriors for each cluster:

$$c_1: p(c_1 = 1|x_3) = \frac{p(c_1 = 1, x_3)}{p(c_1 = 1, x_3) + p(c_2 = 1, x_3)} = 0.9827144048774182$$
$$c_2: p(c_2 = 1|x_3) = \frac{p(c_2 = 1, x_3)}{p(c_1 = 1, x_3) + p(c_2 = 1, x_3)} = 0.01728559512258177$$

- For x_4:

- For cluster c_1:
  * Prior: $p(c_1 = 1) = \pi_1 = 0.7$
  * Likelihood: $p(x_4|c_1 = 1) = \frac{1}{2\pi}\frac{1}{\sqrt{det(\Sigma_1)}} exp\left(-\frac{1}{2}(x_4 - \mu_1)^T(\Sigma_1{}^{-1})(x_4 - \mu_1)\right) = 7.2256232377243294 \times 10^{-6}$
  * Joint Probability: $p(c_1 = 1, x_4) = p(c_1 = 1)p(x_4|c_1 = 1) = 5.05793626640703 \times 10^{-6}$
- For cluster c_2:
  * Prior: $p(c_2 = 1) = \pi_2 = 0.3$
  * Likelihood: $p(x_4|c_2 = 1) = 2.8136605178593184 \times 10^{-6}$
  * Joint Probability: $p(c_2 = 1, x_4) = p(c_2 = 1)p(x_4|c_2 = 1) = 8.440981553577955 \times 10^{-7}$
- So, we can compute the normalized posteriors for each cluster:

$$c_1: p(c_1 = 1|x_4) = \frac{p(c_1 = 1, x_4)}{p(c_1 = 1, x_4) + p(c_2 = 1, x_4)} = 0.8569818311724802$$
$$c_2: p(c_2 = 1|x_4) = \frac{p(c_2 = 1, x_4)}{p(c_1 = 1, x_4) + p(c_2 = 1, x_4)} = 0.1430181688275199$$

M-Step: Re-estimate cluster parameters such that they ft their assigned elements.

For each cluster we need to compute the new prior and likelihood parameters (centroids and covariance matrices), given by:

$$\mu_c = \frac{\sum_{n=1}^{4} p(c_c = 1|x_n) \times x_n}{\sum_{n=1}^{4} p(c_c = 1|x_n)} \qquad \Sigma_{ij}^{c} = \frac{\sum_{n=1}^{4} p(c_c = 1|x_n) \times (x_{ni} - \mu_{ci})(x_{nj} - \mu_{cj})}{\sum_{n=1}^{4} p(c_c = 1|x_n)} \qquad p(C = c) = \frac{\sum_{n=1}^{4} p(c_c = 1|x_n)}{\sum_{l=1}^{2}\sum_{n=1}^{4} p(c_l = 1|x_n)}$$

, where x_{ni} represents feature y_i of observation x_n and μ_{ci} represents index i of centroid μ_c.

So, let us estimate the new parameters for each cluster.

- For cluster c_1:
  - For the likelihood:

$\mu_1 = \begin{pmatrix} 1.56538325 \\ 2.10072779 \end{pmatrix}$

$\Sigma_{11}^1 = 4.132822983777583$

$\Sigma_{12}^1 = \Sigma_{21}^1 = -1.1633677940727911$

$\Sigma_{22}^1 = 2.6056010565143386$

  - For the prior:

$p(c_1 = 1) = 0.7099240583770576$

- For cluster c_2:
  - For the likelihood:

$\mu_2 = \begin{pmatrix} -0.38370376 \\ -3.41757815 \end{pmatrix}$

$\Sigma_{11}^2 = 2.701660141860462$

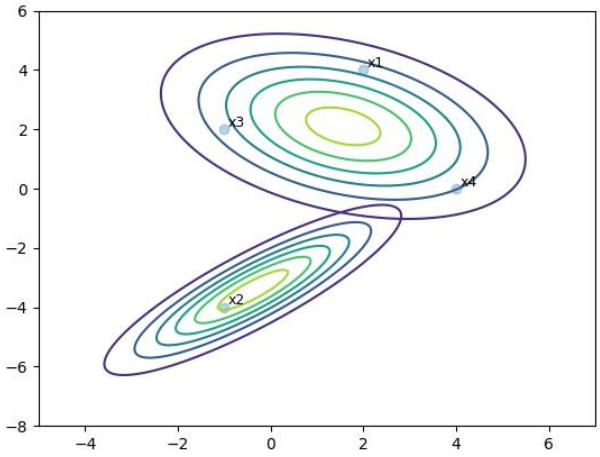
$\Sigma_{12}^2 = \Sigma_{21}^2 = 2.106240599436862$

$\Sigma_{22}^2 = 2.169241946340971$

  - For the prior:

$p(c_2 = 1) = 0.29007594162294237$

The sketch of the points of the solution is given by:



2) Although in EM we allow non-deterministic distributions, i.e., a point does not necessarily belong to one cluster, but instead it's a member of every cluster with different degrees of probability, for the silhouette coefficient computation process we assumed that x_i belonged to c_1 if $p(c_1 = 1|x_i) > p(c_2 = 1|x_i)$ and vice-versa. Therefore, x_1, x_3 and $x_4 \in c_1$ and $x_2 \in c_2$.

Preserving the Euclidean space distance assumption, let us compute the silhouette of c_1:

$$s(x_1) = 1 - \frac{a(x_1)}{b(x_1)} = 1 - \frac{\frac{\|x_1 - x_3\|_2 + \|x_1 - x_4\|_2}{2}}{\|x_1 - x_2\|_2} = 1 - \frac{\frac{\sqrt{(2+1)^2 + (4-2)^2} + \sqrt{(2-4)^2 + (4-0)^2}}{2}}{\sqrt{(2+1)^2 + (4+4)^2}} = 0.52728$$

$$s(x_3) = 1 - \frac{a(x_3)}{b(x_3)} = 1 - \frac{\frac{\|x_3 - x_1\|_2 + \|x_3 - x_4\|_2}{2}}{\|x_3 - x_2\|_2} = 1 - \frac{\frac{\sqrt{(-1-2)^2 + (2-4)^2} + \sqrt{(-1-4)^2 + (2-0)^2}}{2}}{\sqrt{(-1+1)^2 + (2+4)^2}} = 0.25077$$

$$s(x_4) = 1 - \frac{a(x_4)}{b(x_4)} = 1 - \frac{\frac{\|x_4 - x_1\|_2 + \|x_4 - x_3\|_2}{2}}{\|x_4 - x_2\|_2} = 1 - \frac{\frac{\sqrt{(4-2)^2 + (0-4)^2} + \sqrt{(4+1)^2 + (0-2)^2}}{2}}{\sqrt{(4+1)^2 + (0+4)^2}} = 0.23027$$

Therefore,

$$s(c_1) = \frac{s(x_1) + s(x_3) + s(x_4)}{3} = \frac{0.52728 + 0.25077 + 0.23027}{3} = 0.33610$$

Since point x_2 is the only one that belongs to c_2, $a(x_2) = 0$. Therefore,

$$s(c_2) = s(x_2) = 1 - \frac{0}{b(x_2)} = 1$$

The silhouette of a solution is the average of cluster silhouettes.

$$silhouette(C) = \frac{s(c_1) + s(c_2)}{2} = 0.66805$$

The silhouette score indicates that the clustering solution is reasonable but not too accurate. From looking at the sketch that is because the cluster 1 is not very cohesive. There would be needed more iterations to find a more fitting EM clustering solution.

3) (a) We know that the VC dimension is a measure of the degrees of freedom of a classifier. A good proxy for the number of degrees of freedom is the number of parameters. So, let us estimate the number of parameters for each scenario.

i. Between the input layer and the first hidden layer we will have:

- A 5×5 weight matrix $\boldsymbol{W}^{[1]}$, so 5^2 parameters
- A 5×1 bias vector $\boldsymbol{b}^{[1]}$, so 5 parameters

Since we have two more hidden, both with 5 nodes, we have two more sets of $5^2 + 5$ parameters.

Between the third hidden layer and the output layer we will have:

- A 2×5 weight matrix $\boldsymbol{W}^{[4]}$, so 10 parameters.
- A 2×1 bias vector $\boldsymbol{b}^{[4]}$, so 2 parameters.

In total, we have $3 \times (5^2 + 5) + (10 + 2) = 102$ parameters.

ii. If input variables are discretized using three bins, we have that 3^d is the maximum number of different points that can exist in this scenario, where d is the data dimensionality. Therefore $d_{VC} \leq 3^d$ for any classifier with this kind of input. Noticing that, all we need is to create a decision tree that can shatter a dataset with all possible points. Using simple ternary splits on all features, we can create a decision tree with height $d + 1$ that has one leaf for each point. Each leaf will have the label of that point. So, all points will be classified correctly regardless of the chosen label. This proves that $d_{VC} \geq 3^d$. Therefore $d_{VC} = 3^d$. Since $d = 5$, the VC dimension of a decision tree assuming input variables are discretized using three bins is $3^5 = 243$.

iii. For a binary Bayesian classifier with a multivariate Gaussian likelihood, we need to estimate prior and likelihood.

$p(z = 0) = p$ and $p(z = 1) = 1 - p$, so we have 1 parameter for the prior.

The likelihood $p(x|z = 0)$, being a multivariate gaussian, will require a mean vector and a covariance matrix. For a data dimensionality of size 5:

- The mean vector is 5×1, so 5 parameters.
- The covariance matrix is a 5×5 symmetric matrix, so we only need to count the diagonal and upper part of matrix. So, $\frac{5 \times 6}{2} = 15$ parameters.

The likelihood $p(x|z = 1)$, being a multivariate gaussian, requires the same number of parameters.

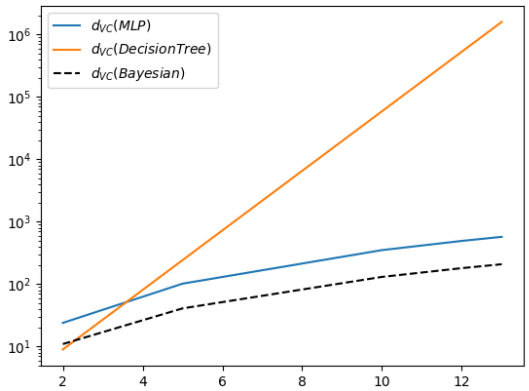
There are no more parameters to estimate, so the total is $1 + 2 \times (5 + 15) = 41$ parameters.

TÉCNICO LISBOA

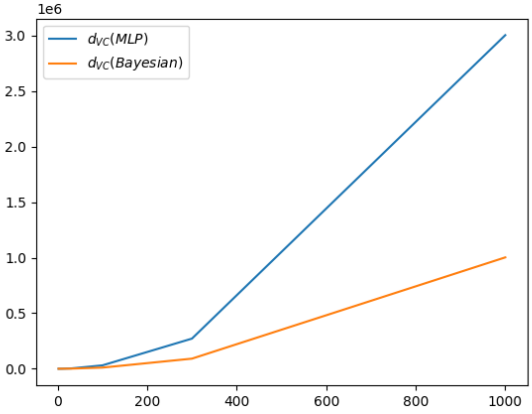
Aprendizagem 2021/22

Homework I – Group 053

(b) As we can see, (i) and (iii) are both quadratic with different coefficients yet both have a very similar growth and are almost asymptotically undistinguishable when compared to (ii) which requires a bigger overhead with its dimensionality exponential on the problem's data dimensionality, leading us to empirically conclude that approximately $d_{VC}(Bayesian) < d_{VC}(MLP) < d_{VC}(DecisionTree)$.



(c) Generalizing the VC-dimension estimate for data dimensionality m we have that (i) is given by $m^2 + 3m + 1$ and (iii) by $3m^2 + 5m + 2$, where we computed that (iii)'s slope is approximately 3 times bigger ($\lim_{m\to\infty} \frac{3m^2+5m+2}{m^2+3m+1} = 3$) than (i)'s, leading us to conclude that approximately $d_{VC}(Bayesian) < d_{VC}(MLP)$.



II. Programming and critical analysis

4) a. The Error Classification Rate (ECR) is given by: $ECR = \frac{1}{K}\sum_{k=1}^{K}(|c_k| - \varphi_L(c_k))$, where $\varphi_L(c_k) = \max_{j=1..G}(|c_k \cap L_j|)$ and $C = \{c_1, c_2, \ldots, c_k\}, L = \{L_1, L_2, \ldots, L_G\}$ are the sets of clusters and reference classes, respectively.

$k = 2$:

$$ECR = \frac{1}{2}((453 - 435) + (230 - 221)) = 13.5$$

$k = 3$:

$$ECR = \frac{1}{3}((442 - 433) + (126 - 126) + (115 - 104)) = 6.667$$

Therefore, $ECR(k = 2) > ECR(k = 3)$.

b.

$k = 2$:

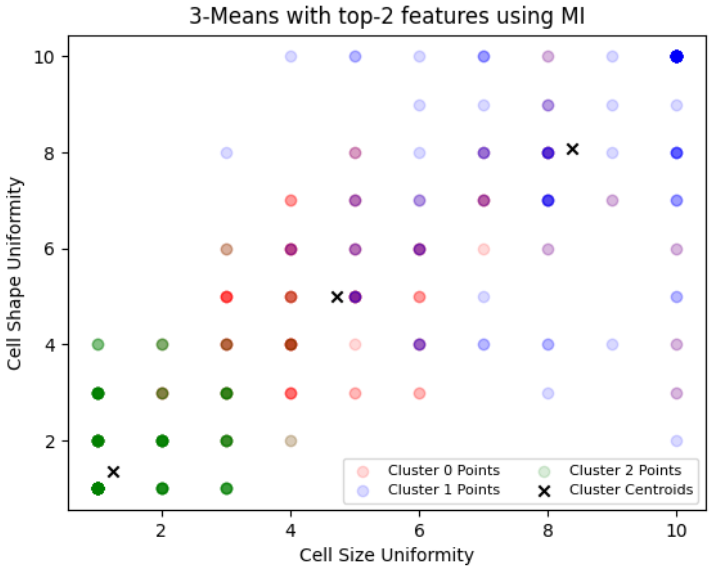
$Silhouette\ Score = 0.5967981179111456$

$k = 3$:

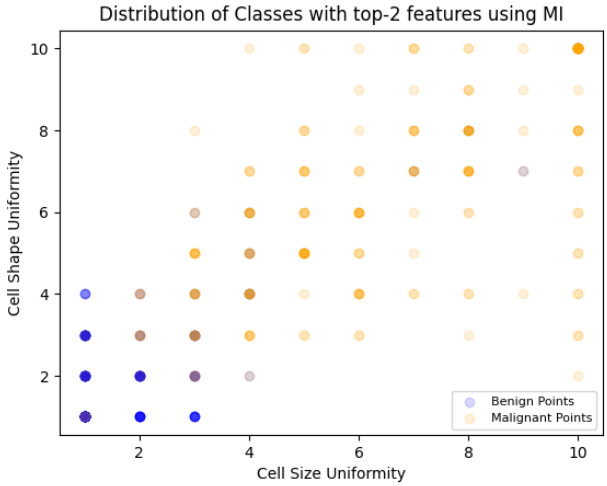
$Silhouette\ Score = 0.5245427800706391$

Therefore, $Silhouette\ Score(k = 2) > Silhouette\ Score(k = 3)$.

5) The top-2 features with higher mutual information are Cell Size Uniformity and Cell Shape Uniformity. Let us plot the k=3 clustering solution using these features:



6)



k=3 clustering has a lower ECR when compared to k=2, which is visible since the data points are not uniformly distributed – there is a bias in benign diagnostic towards small values of Cell Size Uniformity and Cell Shape Uniformity (blue points). The malignant results spread over a bigger range of values, therefore having 2 out of 3 clusters representing those cases and that allows a better interpretation of the data. The silhouette score is moderate because there is some cohesion amongst the points of each cluster but there is few to no separation between those of different ones. Consequently, based on the evaluation of the clustering options, using internal and external criteria (Silhouette and ECR, respectively), k=3 would achieve a better diagnostic accuracy.

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 053

III. APPENDIX

```
# Ex. 1

df = pd.read_csv(r'dataset.csv')
print(df.to_string(index=False))

prior_1 = 0.7
prior_2 = 0.3
centroid_1 = np.array(df.iloc[0].values)
centroid_2 = np.array(df.iloc[1].values)
cov_1 = np.matrix([[1,0], [0,1]])
cov_2 = np.matrix([[2,0], [0,2]])

joint_prob_c1 = {}
joint_prob_c2 = {}
cluster_1_post = []
cluster_2_post = []

# Cluster 1
print(f'\n{bcolors.OKCYAN}Cluster 1{bcolors.ENDC}\n')
for index, row in df.iterrows():
    obs = [row["Y1"], row["Y2"]]
    multivargauss_likelihood_c1 = stats.multivariate_normal.pdf(obs, mean=centroid_1,
cov=cov_1)
    joint_prob_c1[index] = prior_1 * multivargauss_likelihood_c1
    print(f"Multivar(x{index+1}): {multivargauss_likelihood_c1} || Joint prob(x{index+1}):
{joint_prob_c1[index]}")

# Cluster 2
print(f'\n{bcolors.OKCYAN}Cluster 2{bcolors.ENDC}\n')
for index, row in df.iterrows():
    obs = [row["Y1"], row["Y2"]]
    multivargauss_likelihood_c2 = stats.multivariate_normal.pdf(obs, mean=centroid_2,
cov=cov_2)
    joint_prob_c2[index] = prior_2 * multivargauss_likelihood_c2
    print(f"Multivar(x{index+1}): {multivargauss_likelihood_c2} || Joint prob(x{index+1}):
{joint_prob_c2[index]}")

for i in range(4):
    cluster_1_post += [joint_prob_c1[i] / (joint_prob_c1[i] + joint_prob_c2[i])]
    cluster_2_post += [joint_prob_c2[i] / (joint_prob_c1[i] + joint_prob_c2[i])]
    print(f'p(c=1 | x{i+1}) = {cluster_1_post[i]}')
    print(f'p(c=2 | x{i+1}) = {cluster_2_post[i]}')
```

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 053

```
new_centroid_1 = np.multiply((np.multiply(np.array(df.iloc[0].values), cluster_1_post[0])
+ np.multiply(np.array(df.iloc[1].values), cluster_1_post[1]) +
np.multiply(np.array(df.iloc[2].values), cluster_1_post[2]) +
np.multiply(np.array(df.iloc[3].values), cluster_1_post[3])), (1/sum(cluster_1_post)))
new_cov11_1 = ((cluster_1_post[0] * (np.array(df.iloc[0].values)[0] - new_centroid_1[0]) *
(np.array(df.iloc[0].values)[0] - new_centroid_1[0])) + (cluster_1_post[1] *
(np.array(df.iloc[1].values)[0] - new_centroid_1[0]) * (np.array(df.iloc[1].values)[0] -
new_centroid_1[0])) + (cluster_1_post[2] * (np.array(df.iloc[2].values)[0] -
new_centroid_1[0]) * (np.array(df.iloc[2].values)[0] - new_centroid_1[0])) +
(cluster_1_post[3] * (np.array(df.iloc[3].values)[0] - new_centroid_1[0]) *
(np.array(df.iloc[3].values)[0] - new_centroid_1[0]))) / sum(cluster_1_post)
new_cov12_1 = ((cluster_1_post[0] * (np.array(df.iloc[0].values)[0] - new_centroid_1[0]) *
(np.array(df.iloc[0].values)[1] - new_centroid_1[1])) + (cluster_1_post[1] *
(np.array(df.iloc[1].values)[0] - new_centroid_1[0]) * (np.array(df.iloc[1].values)[1] -
new_centroid_1[1])) + (cluster_1_post[2] * (np.array(df.iloc[2].values)[0] -
new_centroid_1[0]) * (np.array(df.iloc[2].values)[1] - new_centroid_1[1])) +
(cluster_1_post[3] * (np.array(df.iloc[3].values)[0] - new_centroid_1[0]) *
(np.array(df.iloc[3].values)[1] - new_centroid_1[1]))) / sum(cluster_1_post)
new_cov22_1 = ((cluster_1_post[0] * (np.array(df.iloc[0].values)[1] - new_centroid_1[1]) *
(np.array(df.iloc[0].values)[1] - new_centroid_1[1])) + (cluster_1_post[1] *
(np.array(df.iloc[1].values)[1] - new_centroid_1[1]) * (np.array(df.iloc[1].values)[1] -
new_centroid_1[1])) + (cluster_1_post[2] * (np.array(df.iloc[2].values)[1] -
new_centroid_1[1]) * (np.array(df.iloc[2].values)[1] - new_centroid_1[1])) +
(cluster_1_post[3] * (np.array(df.iloc[3].values)[1] - new_centroid_1[1]) *
(np.array(df.iloc[3].values)[1] - new_centroid_1[1]))) / sum(cluster_1_post)
new_prior_1 = sum(cluster_1_post) / (sum(cluster_1_post) + sum(cluster_2_post))

new_centroid_2 = np.multiply((np.multiply(np.array(df.iloc[0].values), cluster_2_post[0])
+ np.multiply(np.array(df.iloc[1].values), cluster_2_post[1]) +
np.multiply(np.array(df.iloc[2].values), cluster_2_post[2]) +
np.multiply(np.array(df.iloc[3].values), cluster_2_post[3])), (1/sum(cluster_2_post)))
new_prior_2 = sum(cluster_2_post) / (sum(cluster_1_post) + sum(cluster_2_post))
new_cov11_2 = ((cluster_2_post[0] * (np.array(df.iloc[0].values)[0] - new_centroid_2[0]) *
(np.array(df.iloc[0].values)[0] - new_centroid_2[0])) + (cluster_2_post[1] *
(np.array(df.iloc[1].values)[0] - new_centroid_2[0]) * (np.array(df.iloc[1].values)[0] -
new_centroid_2[0])) + (cluster_2_post[2] * (np.array(df.iloc[2].values)[0] -
new_centroid_2[0]) * (np.array(df.iloc[2].values)[0] - new_centroid_2[0])) +
(cluster_2_post[3] * (np.array(df.iloc[3].values)[0] - new_centroid_2[0]) *
(np.array(df.iloc[3].values)[0] - new_centroid_2[0]))) / sum(cluster_2_post)
new_cov12_2 = ((cluster_2_post[0] * (np.array(df.iloc[0].values)[0] -
new_centroid_2[0]) * (np.array(df.iloc[0].values)[1] - new_centroid_2[1])) +
(cluster_2_post[1] * (np.array(df.iloc[1].values)[0] - new_centroid_2[0]) *
(np.array(df.iloc[1].values)[1] - new_centroid_2[1])) + (cluster_2_post[2] *
(np.array(df.iloc[2].values)[0] - new_centroid_2[0]) *
(np.array(df.iloc[2].values)[1] - new_centroid_2[1])) + (cluster_2_post[3] *
```

```
(np.array(df.iloc[3].values)[0] - new_centroid_2[0]) * (np.array(df.iloc[3].values)[1] -
new_centroid_2[1]))) / sum(cluster_2_post)
new_cov22_2 = ((cluster_2_post[0] * (np.array(df.iloc[0].values)[1] - new_centroid_2[1]) *
(np.array(df.iloc[0].values)[1] - new_centroid_2[1])) + (cluster_2_post[1] *
(np.array(df.iloc[1].values)[1] - new_centroid_2[1]) * (np.array(df.iloc[1].values)[1] -
new_centroid_2[1])) + (cluster_2_post[2] * (np.array(df.iloc[2].values)[1] -
new_centroid_2[1]) * (np.array(df.iloc[2].values)[1] - new_centroid_2[1])) +
(cluster_2_post[3] * (np.array(df.iloc[3].values)[1] - new_centroid_2[1]) *
(np.array(df.iloc[3].values)[1] - new_centroid_2[1]))) / sum(cluster_2_post)

# Sketch the solution
N=500
X=np.linspace(-5,7,N)
Y=np.linspace(-8,6,N)
X, Y=np.meshgrid(X, Y)
pos=np.dstack((X,Y))
rv=stats.multivariate_normal(new_centroid_1, np.matrix([[new_cov11_1, new_cov12_1],
[new_cov12_1, new_cov22_1]]))
Z = rv.pdf(pos)
plt.contour(X,Y,Z)

rv_2=stats.multivariate_normal(new_centroid_2, np.matrix([[new_cov11_2, new_cov12_2],
[new_cov12_2, new_cov22_2]]))
Z_2 = rv_2.pdf(pos)
plt.contour(X,Y,Z_2)

y1 = np.array(df['Y1'].values)
y2 = np.array(df['Y2'].values)
plt.scatter(y1, y2, alpha=0.3)
for i in range(len(y1)):
    plt.text(y1[i]+0.08, y2[i]+0.08, f"x{i+1}", fontsize=9)
plt.savefig('ex1.png')

# Ex. 3
m_plot = np.array([2,5,10,12,13])

mlp = 3 * m_plot ** 2 + 5 * m_plot + 2
dt = 3 ** m_plot
bayesian_mvg = m_plot ** 2 + 3 * m_plot + 1

plt.plot(m_plot, mlp, label=r'$d_{VC}(MLP)$')
plt.plot(m_plot, dt, label=r'$d_{VC}(DecisionTree)$')
plt.plot(m_plot, bayesian_mvg, label=r'$d_{VC}(Bayesian)$', linestyle='--', color='black')
plt.yscale('log')
plt.legend()
plt.savefig('ex3b.png')
```

```
plt.clf()

plt.plot(m_plot, mlp, label=r'$d_{VC}(MLP)$')
plt.plot(m_plot, bayesian_mvg, label=r'$d_{VC}(Bayesian)$')
plt.legend()
plt.savefig('ex3c.png')

# Exs. 4, 5, 6
data = arff.loadarff('breast.w.arff')
df = pd.DataFrame(data[0])
df['Class'] = df['Class'].str.decode('utf8') # remove weird string
X = df.iloc[:,:-1].values.tolist()
y = df.iloc[:,-1].values.tolist()

# Ex.4
ecr = []
sil = []
y_pred = []
kmeans = []

def phil_ck(y_pred, k):
    lens_indexes = []
    max_factor = []
    for i in range(k):
        benign_count, malignant_count = 0, 0
        index = [a for a, b in enumerate(y_pred) if b == i]
        for j in range(len(index)):
            if df.iloc[index[j]]['Class'] == 'benign':
                benign_count += 1
            else:
                malignant_count += 1
        lens_indexes.append(len(index))
        max_factor.append(max(benign_count, malignant_count))
    return lens_indexes, max_factor

def ecr_score(X, y_pred, k):
    lens, maxs = phil_ck(y_pred, k)
    print(lens, maxs)
    return (1/k) * sum([a - b for a, b in zip(lens, maxs)])
for i in range(2, 4):
    kmeans.append(KMeans(n_clusters=i, random_state=0))
    y_pred.append(kmeans[i-2].fit_predict(X))
    ecr.append((i, ecr_score(X, y_pred[i-2], i)))
    sil.append((i, silhouette_score(X, y_pred[i-2])))

print(f'ECR: {ecr}')
```

```
print(f'SIL: {sil}')

# Ex. 5
selector = SelectKBest(mutual_info_classif, k=2)
X_new = selector.fit_transform(X, y)
cols = selector.get_support(indices=True)
features_df_new = df.iloc[:,cols]
centroids = kmeans[1].cluster_centers_[:,cols]
index_0 = [a for a, b in enumerate(y_pred[1]) if b == 0]
index_1 = [a for a, b in enumerate(y_pred[1]) if b == 1]
index_2 = [a for a, b in enumerate(y_pred[1]) if b == 2]
cl_0 = plt.scatter([X_new[index_0][:, 0]], X_new[index_0][:, 1], color='r', alpha=0.15)
cl_1 = plt.scatter([X_new[index_1][:, 0]], X_new[index_1][:, 1], color='b', alpha=0.15)
cl_2 = plt.scatter([X_new[index_2][:, 0]], X_new[index_2][:, 1], color='g', alpha=0.15)
centroids = plt.scatter(centroids[:,0], centroids[:,1], marker='x', color='black')
plt.xlabel("Cell Size Uniformity")
plt.ylabel("Cell Shape Uniformity")
plt.title(f"3-Means with top-2 features using MI")
plt.legend((cl_0, cl_1, cl_2, centroids),
           ('Cluster 0 Points', 'Cluster 1 Points', 'Cluster 2 Points', 'Cluster
Centroids'),
           scatterpoints=1,
           loc='lower right',
           ncol=2,
           fontsize=8)
plt.savefig('ex5.png')
plt.clf()

#Ex. 6
benign = df[df['Class'] == 'benign'][['Cell_Size_Uniformity',
'Cell_Shape_Uniformity']].to_numpy()
malignant = df[df['Class'] == 'malignant'][['Cell_Size_Uniformity',
'Cell_Shape_Uniformity']].to_numpy()
ben = plt.scatter([benign[:, 0]], benign[:, 1], color='blue', alpha=0.15)
mal = plt.scatter([malignant[:, 0]], malignant[:, 1], color='orange', alpha=0.15)
plt.xlabel("Cell Size Uniformity")
plt.ylabel("Cell Shape Uniformity")
plt.title("Distribution of Classes with top-2 features using MI")
plt.legend((ben, mal),
           ('Benign Points', 'Malignant Points'),
           scatterpoints=1,
           loc='lower right',
           fontsize=8)
plt.savefig('aux_ex6.png')
```

END