SB 2.2 Statistical Machine Learning

Statistical Machine Learning

Sheet 1 — HT21

Unsupervised learning

1. Let X be a p-dimensional random vector with zero mean and covariance matrix Σ.

   (a) Under what condition will the first principal component direction be identifiable? (It is not identifiable if there are more than one direction satisfying the defining criterion).

   (b) Supposing it is not identifiable, can you describe the behaviour of the first empirical principal component when computed using a dataset $(x_1, \ldots, x_n)$, then on $(x_1, \ldots, x_{n+m})$, $m \geq 1$?
   [*hint: what happens when PCA is applied to samples from an isotropic Gaussian?*]

2. We perform PCA on a centered dataset $(x_1, \ldots, x_n)$. Let $S = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^\intercal$ be the sample covariance. Denote the projection of x_i onto the jth (empirical) principal component by z_{ij}.

   (a) Find the sample mean and sample variance of the projections on the jth PC $(z_{1j}, \ldots, z_{nj})$

   (b) Show that the total sample variance of $(x_1, \ldots, x_n)$ is equal to the sum of the sample variances of the PC projections.

3. Let $(x_1, \ldots, x_n)$ be a centered dataset, where $x_i \in \mathbb{R}^p$. For a p-by-K matrix A with orthonormal columns, that is $A^\intercal A = I_K$, consider the linear encoder and decoder defined by

$$z_i = \mathrm{enc}_A(x_i) = A^\intercal x_i \tag{1}$$

$$\widehat{x}_i = \mathrm{dec}_A(z_i) = A z_i \tag{2}$$

   Let $h_A(x_i) = AA^\intercal x_i = \widehat{x}_i$ be the corresponding linear autoencoder.

   (a) Give the empirical risk $\widehat{R}_n(h_A)$ of the autoencoder h_A under a squared loss, as a function of $(x_i)_{i=1,\ldots,n}$, A and n.

   (b) Show that $\widehat{R}_n(h_A) = \frac{n-1}{n} \left(\mathrm{trace}(S) - \mathrm{trace}(A^\intercal S A)\right)$ where S is the sample covariance matrix.

   (c) Let $v_1, \ldots, v_p$ be the eigenvectors and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$ be the eigenvalues of S. Show that the p-by-K matrix $V_{1:K} = (v_1, \ldots, v_K)$ is the empirical risk minimiser amongst the class of linear autoencoders h_A, under a squared loss, and give the expression for $\widehat{R}_n(V_{1:K})$.

François Caron: caron@stats.ox.ac.uk

4. (a) Under the assumption that your data $(x_1, \dots, x_n)$ are centered, show that you can compute the $n \times n$ Gram matrix B such that $b_{ij} = x_i^\intercal x_j$ using the dissimilarity matrix A where $a_{ij} = \|x_i - x_j\|$.

   (b) Assume that you only observe pairwise euclidian distances a_{ij} between pairs of observations (not the raw data $(x_1, \dots, x_n)$). Explain how you can obtain the PCA projections. Can you recover the principal components?

5. Let $x_1, \dots, x_n$ be a dataset where $x_i \in \mathbb{R}^p$ and $\Pi = \{C_1, C_2, \dots, C_K\}$ a partition of $\{1, \dots, n\}$. For each cluster C_k, denote $n_k = |C_k|$ and define

$$\bar{x}_k = \frac{1}{n_k} \sum_{i \in C_k} x_i \qquad \text{to be the within-cluster sample mean}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^{K} n_k \bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \text{to be the overall sample mean}$$

   Let

$$\widetilde{W}(C_1, \dots, C_n) = \frac{1}{2n} \sum_{i, i' \in C_k} \|x_i - x_{i'}\|^2$$

   be the K-means objective function.

   (a) Show that

$$\widetilde{W}(C_1, \dots, C_n) = \sum_{i \in C_k} \|x_i - \bar{x}_k\|^2$$

   (b) Let

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^\top \qquad \text{be the sample covariance matrix}$$

$$W = \frac{1}{n-1} \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^\top \qquad \text{to be the sample within-cluster covariance matrix}$$

$$B = \frac{1}{n-1} \sum_{k=1}^{K} n_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^\top \qquad \text{to be the sample between-cluster covariance matrix}$$

   where S, W and B are all $p \times p$ matrices. Show that $S = W + B$

   (c) Explain how the K-means objective function $\widetilde{W}$ is related to the matrix W.

   (d) Let $z_i = k$ if $i \in C_k$. For $i = 1, \dots, n$, write $\widetilde{x}_i = \bar{x}_{z_i}$. Show that the total sample variance $\widetilde{T}$ of $\widetilde{x}_1, \dots, \widetilde{x}_n$ is upper bounded by the total sample variance T of $(x_1, \dots, x_n)$. Show that minimising $\widetilde{W}$ is equivalent to maximising $\widetilde{T}$.

François Caron: caron@stats.ox.ac.uk

6. The following Python code loads the crabs datasets into a data frame $\mathbf{X}$.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from scipy.linalg import eigh # eigendecomposition
from numpy.linalg import svd # SVD

# Load the crabs dataset
url = 'https://vincentarelbundock.github.io/Rdatasets/csv/MASS/crabs.csv'
crabs = pd.read_csv(url)
X = crabs[['FL', 'RW', 'CL', 'CW', 'BD']]
```

Using Python (recommended) or R:

(a) Compute the sample covariance S

(b) Compute the eigenvalues and the principal components using the eigendecomposition of S

(c) Compute the PCA projections, and plot them using a pairplot (function pairplot in seaborn)

(d) Compute the eigenvalues and the principal components using the SVD decomposition of $\mathbf{X}$

(e) Compute the Gram matrix B

(f) Compute the PCA projections from the eigendecomposition of B

François Caron: caron@stats.ox.ac.uk

7. (Optional) Spike sorting is an important problem in computational neuroscience. Based on recordings of neuronal activity, it aims at detecting spikes, and assigning each spike to the activity of a given neuron. The dataset we will consider is composed of $n = 1000$ spike recordings. For each spike, a set of $p = 96$ features have been extracted using some preprocessing tools. The data[1] are stored in the csv file spike_data.csv. Here is the code to load the data as a panda frame.

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
X = pd.read_csv('spike_data.csv')
```

Using Python (recommended) or R:

(a) Apply PCA to the data X, and compute the projections of the data onto the first two components.

(b) Apply Kmeans with two clusters to the PCA projections, and visualise the obtained partition.

(c) Let $\widehat{\mu}_1 \in \mathbb{R}^2$ and $\widehat{\mu}_2 \in \mathbb{R}^2$ be the estimated cluster means. Map these means into the original data space of dimension 96 using the PCA decoder. Plot the two transformed cluster means.

[1] This is a subset of the data available from: https://ifcs.boku.ac.at/repository/data/spike_sorting/index.html

François Caron: caron@stats.ox.ac.uk

SB2.2 Statistical Machine Learning Sheet 1 Ruizhen Ma.

1. (a) The first PC is indentifiable if $\exists$ largest eigenvalue λ_1 of Σ s.t. for all eigenvalues of Σ, $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$.

(b) If it's not identifiable, this implies the dimension of the eigenspace for the largest eigenvalue λ_1 is > 1. Therefore there's infinite directions within the eigenspace that maximise $\text{Var}(z_1) = a_1^T S a_1$.
Hence the projection of $(x_1, \cdots x_n)$ and $(x_1, \cdots x_{n+m})$ may be significantly different. $\Rightarrow$ unstable

2. (a) $z_{ij} = v_j^T x_i$

Sample mean: $\frac{1}{n}\sum z_{ij} = \frac{1}{n}\sum v_j^T x_i = \frac{1}{n} v_j^T \underbrace{\sum x_i}_{0} = 0$

Sample variance: $\frac{1}{n-1}\sum z_{ij}^2 = \frac{1}{n-1}\sum v_j^T x_i x_i^T v_j = v_j^T S v_j = \lambda_j$

$= v_j^T (\frac{1}{n-1} x_i x_i^T) v_j = v_j^T S v_j = \lambda_j v_j^T v_j = \lambda_j$

(b) $TSV(X) = \frac{1}{n-1}\sum_i \sum_j x_{ij}^2$

$= Tr(S) = Tr(V \Lambda V^T) = Tr(\Lambda) = \sum \lambda_j$ □

$\sum_{j=1}^{p} S_{jj} = \text{trace}(S) \quad S = V \Lambda V^T$

$= \text{trace}(V \Lambda V^T)$

$= \text{trace}(\Lambda V^T V) = \text{trace}(\Lambda)$

$= \sum \lambda_j$

3. (a) Empirical Risk:

$$\hat{R}_n(h_A) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - AA^T x_{ij})^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|x_i - AA^T x_i\|^2$$

(b) Proof: $\hat{R}_n(h_A) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - AA^T x_{ij})^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}[(I_k - AA^T)x_{ij}]^2$

$\Rightarrow \hat{R}_n(h_A) = TV((I_k - AA^T)X)\cdot\frac{n-1}{n}$, where TV is the total sample variance.

Why? You did not use $A^TA = I_k$.

$$= \frac{n-1}{n}TV(X) - \frac{n-1}{n}TV(AA^TX)$$

$$= \frac{n-1}{n}[Trace(S) - Trace(AA^TS)]$$

$$= \frac{n-1}{n}[Trace(S) - Trace(A^TSA)]$$

(c) From (b), given an encoder A^T, the empirical risk is:

$$\hat{R}_n(h_A) = \frac{n-1}{n}[Trace(S) - Trace(A^TSA)]$$

We seek for $V_{1:k}$ s.t.

$$V_{1:k} = \underset{V_{1:k}\in A}{\arg\min}\ \hat{R}_n(h_A) \iff \underset{V_{1:k}\in A}{\arg\max}\ Trace(A^TSA)$$

Since $v_1, \dots v_p$ be the eigenvectors and $\lambda_1 \geq \lambda_2 \dots \geq 0$ be the eigenvalues of S, by eigen-decomposition, $S = V\Lambda V^T$ where $VV^T = I$,

$\Lambda = diag(\lambda_1 \dots \lambda_n)$, $V = (v_1, \dots v_k)$

Therefore $Trace(A^TSA) = Trace(A^TV\Lambda V^TA)$

this is maximized when $A = V_{1:k}$ Why?

Hence $\hat{R}_n(V_{1:k}) = \frac{n-1}{n}[Trace(S) - \sum_{i=1}^{n}\lambda_i]$

$= \dots$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i - AA^Tx_i)^T(x_i - AA^Tx_i)$$

$$= \frac{1}{n}\sum_{i=1}^{n}[x_i^Tx_i - 2x_i^TAA^Tx_i + x_i^TAA^TAA^Tx_i]$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i^Tx_i - x_i^TAA^Tx_i)$$

$AA^T = I_k$.

$$\sum_{i=1}^{n}x_i^Tx_i = trace(\sum_{i=1}^{n}x_i^Tx_i) = \sum_{i=1}^{n}trace(x_i^Tx_i)$$

$$= \sum_{i=1}^{n}trace(x_ix_i^T)$$

$$= (n-1)\,trace(S)$$

$$\sum_{i=1}^{n} x_i^T A A^T x_i = \sum_{i=1}^{n} \text{trace}(x_i^T A A^T x_i)$$

$$= \sum_{i=1}^{n} \text{trace}(A^T x_i x_i^T A) = (n-1)\,\text{trace}(A^T S A)$$

Also $\sum_{i=1}^{n} x_i^T x_i$ $\quad X = \begin{pmatrix} — x_1^T — \\ — x_n^T — \end{pmatrix}$

$= \text{trace}(XX^T)$

$= \text{trace}(X^T X)$ $\quad (XX^T)_{ii} = x_i^T x_i$

$= (n-1)\,\text{trace}(S)$

(c) $\hat{R}_n(h_A) = \frac{n-1}{n}\left[\text{trace}(S) - \text{trace}(A^T S A)\right]$

$\min \hat{R}_n(h_A) \iff \max \text{trace}(A^T S A)$

$= \max \sum_{j=1}^{k} a_j^T S a_j$, where a_j is column of A.

subject to $a_j^T a_j = 1$

By Lagrangian: $L(\lambda) = \sum_{j=1}^{k} a_j^T S a_j - \lambda(a_j^T a_j - 1)$

$\frac{\partial L}{\partial a_j^T} = 2Sa_j - 2\lambda a_j = 0 \Rightarrow a_j$ eigenvectr.

$\Rightarrow V_{1:k} = \underset{\lambda A}{\text{argmax}}\ \text{trace}(A^T S A)$

$\Rightarrow \hat{R}_n(V_{1:k}) = \frac{n-1}{n}\left(\sum_{j=1}^{p} \lambda_j - \sum_{j=1}^{k} \lambda_j\right)$

4. (a) $a_{ij} = \|x_i - x_j\|$, therefore $a_{ij}^2 = \|x_i - x_j\|^2$ b_{ij}

$= \|x_i\|^2 + \|x_j\|^2 - 2x_i^T x_j$

$\Rightarrow \sum_{j=1}^{n} a_{ij}^2 = n\|x_i\|^2 + \sum_{j=1}^{n}\|x_j\|^2 - 2\left(\sum_{j=1}^{n} x_j\right)^T x_i$, $\sum_{j=1}^{n} x_j = 0$

$\Rightarrow \sum_i \sum_j a_{ij}^2 = 2n \sum_{k=1}^{n} \|x_k\|^2$

Hence $\|x_i\|^2 = \frac{1}{n}\left(\sum_{j=1}^{n} a_{ij}^2 - \frac{1}{2n}\sum_i\sum_j a_{ij}^2\right)$

since $b_{ij} = x_i^T x_j$

$= \frac{1}{2}\|x_i\|^2 + \frac{1}{2}\|x_j\|^2 - \frac{1}{2}a_{ij}^2$

by substitution: $b_{ij} = \frac{1}{2n}\left(\sum_{j=1}^{n} a_{ij}^2 - \frac{1}{2n}\sum_i\sum_j a_{ij}^2\right) + \frac{1}{2n}\left(\sum_{i=1}^{n} a_{ij}^2 - \frac{1}{2n}\sum_i\sum_j a_{ij}^2\right) - \frac{1}{2}a_{ij}^2$

$= \frac{1}{2n}\left(\sum_{j=1}^{n} a_{ij}^2 + \sum_{i=1}^{n} a_{ij}^2\right) - \frac{1}{2n^2}\sum_i\sum_j a_{ij}^2 - \frac{1}{2}a_{ij}^2$

(b) Since we only observe $a_{ij} = \|x_i - x_j\|$

We can construct Gram matrix B using (a).

Also, $B = X^T X$, $X = UDV^T$ by singular value decomposition.

PCA Projection: $Z = XV = UDV^TV = UD$.

So Z can be obtained by eigen-decomposition of B.

We can also find V if we have Z and B.

✓ cannot be reconstructed.

$B = -\frac{1}{2}(I - \frac{1}{n}J)\tilde{A}(I - \frac{1}{n}J)$ $\tilde{A}: \tilde{a}_{ij} = a_{ij}^2$

(b) $X = UDV^T$ (Data matrix)

$B = XX^T = UDV^TVD^TU^T = UDDU^T$

$= U\tilde{\Lambda}U^T$

Eigenvectors: U eigenvalue: $\tilde{\Lambda}$

projection: $Z = XV$

$= UD$ $\Rightarrow$ can compute PCA

5. (a) $\tilde{W}(C_1, \cdots C_k) = \frac{1}{2n} \sum_{i,i' \in C_k} \|x_i - x_{i'}\|^2$ (*), $\bar{x}_k = \frac{1}{n_k} \sum_{i \in C_k} x_i$

$$(*) = \frac{1}{2n} \sum_i \sum_{i'} \|x_i - x_{i'}\|^2$$

$$= \frac{1}{2n} \sum_i \sum_{i'} \|x_i - \bar{x}_k + \bar{x}_k - x_{i'}\|^2$$

$$= \frac{1}{2n} \sum_i \sum_{i'} \|x_i - \bar{x}_k\|^2 + \|\bar{x}_k - x_{i'}\|^2 - 2(x_i - \bar{x}_k)^T(\bar{x}_k - x_{i'})$$

$$= \frac{1}{2n} \cdot n \sum_i \|x_i - \bar{x}_k\|^2 + \frac{1}{2n} \cdot n \sum_{i'} \|x_{i'} - \bar{x}_k\|^2 - 2\Big(\sum_i (x_i - \bar{x}_k)\Big)^T \Big(\sum_{i'} (x_{i'} - \bar{x}_k)\Big)$$

$$= \sum_i \|x_i - \bar{x}_k\|^2$$

since $(x_i - \bar{x}_k)^T(\bar{x}_k - x_{i'}) = 0$ by orthogonality ? $\sum_i (x_i - \bar{x}_k) = 0$

(b) Proof: $(n-1)S = \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$

$$= \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - \bar{x}_k + \bar{x}_k - \bar{x})(x_i - \bar{x}_k + \bar{x}_k - \bar{x})^T$$

$$= \sum_{k=1} \sum_{i \in C_k} \big[(x_i - \bar{x}_k)(x_i - \bar{x}_k)^T + (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T + (x_i - \bar{x}_k)(\bar{x}_k - \bar{x})^T + (\bar{x}_k - \bar{x})(x_i - \bar{x}_k)^T \big]$$

$$= \sum_{k=1} \sum_{i \in C_k} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^T + \sum_{k=1} \sum_{i \in C_k} (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T$$

$$+ 0 + 0$$

$$= \sum_{k=1} \sum_{i \in C_k} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^T + \sum_{k=1}^{K} n_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T$$

$$= (n-1)(W + B)$$

Therefore: $S = W + B$

(c) $\tilde{W} = \sum_{i \in C_k} \|x_i - \bar{x}_k\|^2$

$\Rightarrow W = \frac{1}{n-1} \sum_{k=1}^{K} \tilde{W}$ ✗ Use trace

(c) $\tilde{W}(C_1 \cdots C_K) = \sum_{k=1}^{K} \sum_i \|x_i - \bar{x}_k\|^2$

$= (n-1)\,\text{trace}(W)$

(d) $z_i = k$ if $i \in C_k$; $\tilde{x}_i = \bar{x}_{z_i}$

Since

$$\tilde{T} = \frac{1}{n-1}\sum_{k=1}^{K} n_k \|\tilde{x}_i - \bar{x}\|^2 = \frac{1}{n-1}\sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|^2 = B$$

$$T = \frac{1}{n-1}\sum_{i=1}^{n} \|x_i - \bar{x}\|^2 = S$$

by (c) $S = W + B \Rightarrow \tilde{T} \le T$

Since $\tilde{W} = \sum_{k=1}^{K}\sum_{i \in C_k} \|x_i - \bar{x}_k\|^2 = (n-1)W$

$S = W + B$, S is fixed ($B = \tilde{T}$)

Therefore $\arg\min \tilde{W} = \arg\max \tilde{T}$ ✓

$$\text{Trace}(B) = \frac{1}{n-1}\sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|^2$$

$$\frac{1}{n}\sum_{i=1}^{n} \tilde{x}_i = \frac{1}{n}\sum_{k=1}^{K} n_k \bar{x}_k = \bar{x}$$

Total sample variance of $(\hat{x}_1 \cdots \hat{x}_n)$

$$\tilde{T} = \frac{1}{n-1}\sum_{i=1}^{n} \|\tilde{x}_i - \bar{x}\|^2$$

$$= \frac{1}{n-1}\sum_{i=1}^{n} \|\bar{x}_{z_i} - \bar{x}\|^2$$

$$= \frac{1}{n-1}\sum_{k=1}^{n} \|\bar{x}_k - \bar{x}\|^2$$

$$= \text{trace}(B) \le \text{trace}(S)$$

$$\frac{\text{trace}(B)}{\text{trace}(S)} \le 1$$

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from scipy.linalg import eigh # eigendecomposition
from numpy.linalg import svd # SVD
from sklearn.decomposition import PCA

svfigs = False # True if you want to save figures
svdir = './figures/' # directory to save the figures
if svfigs: # set a higher resolution if saving the figures
    plt.rcParams['figure.dpi'] = 150
else:
    plt.rcParams['figure.dpi'] = 75

# Load the crabs dataset
url = 'https://vincentarelbundock.github.io/Rdatasets/csv/MASS/crabs.csv'
crabs = pd.read_csv(url) # load it as a panda table
crabs['class'] = crabs['sp']+ crabs['sex']
crabs = crabs.drop(columns=['Unnamed: 0', 'sp', 'sex', 'index'])

# Data matrix X
X = crabs[['FL', 'RW', 'CL', 'CW', 'BD']]
# Class y (used later on to interpret the results of the PCA)
y = crabs['class']
n, p = X.shape
print('n=' + np.str(n) + ', p='+np.str(p))
X
```

```
n=200, p=5
```

Out[1]:

	FL	RW	CL	CW	BD
0	8.1	6.7	16.1	19.0	7.0
1	8.8	7.7	18.1	20.8	7.4
2	9.2	7.8	19.0	22.4	7.7
3	9.6	7.9	20.1	23.1	8.2
4	9.8	8.0	20.3	23.0	8.2
...	...	...	...	...	...
195	21.4	18.0	41.2	46.2	18.7
196	21.7	17.1	41.7	47.2	19.6
197	21.9	17.2	42.6	47.4	19.5
198	22.5	17.2	43.0	48.7	19.8
199	23.1	20.2	46.2	52.5	21.1

200 rows × 5 columns

In [2]:
```
covMatrix = np.cov(X,bias=False)
```

In [3]:
```
print(covMatrix)
```

```
[[ 33.047   36.654   39.9905 ...  81.1305  83.2585  88.848 ]
 [ 36.654   40.743   44.401  ...  90.006   92.287   98.661 ]
 [ 39.9905  44.401   48.422  ...  98.1145 100.659  107.542 ]
 ...
 [ 81.1305  90.006   98.1145 ... 200.207  205.3015 218.517 ]
```

```
[ 83.2585  92.287  100.659  ... 205.3015 210.633  224.044 ]
[ 88.848   98.661  107.542  ... 218.517  224.044  239.307 ]]
```

In [4]:
```
# Get principal components using scikitlearn
K = 5 # Number of principal components
pca = PCA(n_components = K)
pca.fit(X) #  Calculate the PCs and eigenvalues of the data matrix X

V = pd.DataFrame(data=pca.components_.T, columns=['PC1','PC2','PC3','PC4','PC
print('Principal components:\n', V)

lam = pca.explained_variance_ # eigenvalues
print('\n Eigenvalues/variance of the projections:\n', lam)
```

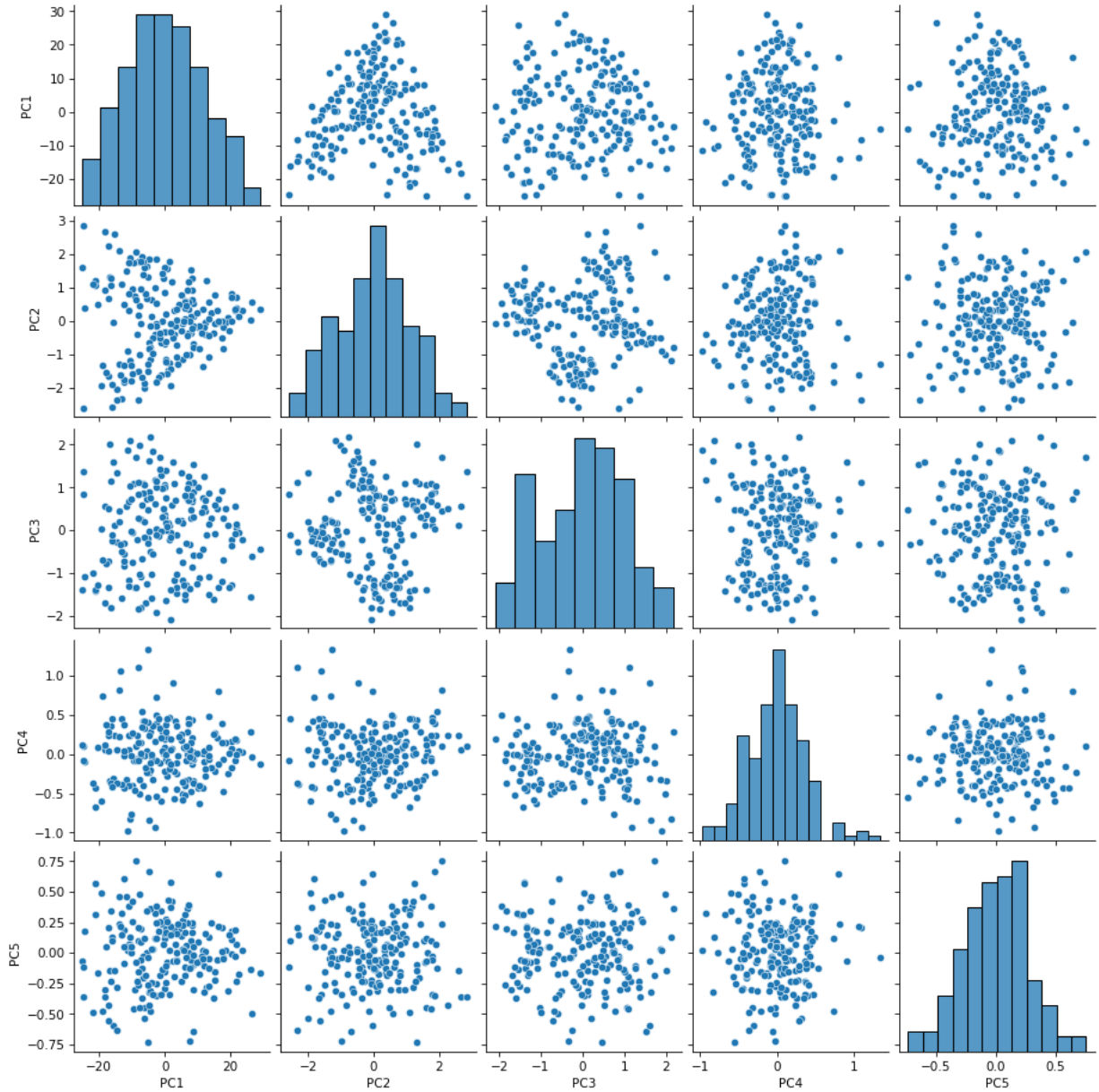
```
Principal components:
          PC1       PC2       PC3       PC4       PC5
FL -0.288981 -0.323250 -0.507170  0.734291 -0.124882
RW -0.197282 -0.864716  0.414136 -0.148309  0.140862
CL -0.599399  0.198226 -0.175330 -0.143594  0.741666
CW -0.661655  0.287979  0.491376  0.125628 -0.471220
BD -0.283732 -0.159845 -0.546882 -0.634366 -0.438687

 Eigenvalues/variance of the projections:
 [1.40705719e+02 1.29683676e+00 1.00026913e+00 1.35299319e-01
 7.79142291e-02]
```

In [5]:
```
# Compute and plot the projections onto the PCs
Z_pca = pca.transform(X) # Projections on the PCs
crabs2 = pd.concat([pd.DataFrame(Z_pca, columns = ['PC1', 'PC2', 'PC3', 'PC4'
print(crabs2.head())

# Pairplot without the classes
plt.figure()
sns.pairplot(data=crabs2)
if svfigs:
    plt.savefig(svdir + 'crabspairplot_pca1.png', bbox_inches = 'tight')
```

```
        PC1       PC2       PC3       PC4       PC5 class
0  26.464575  0.576534 -0.611568 -0.028681 -0.496585    BM
1  23.561737  0.336420 -0.237388  0.022209  0.016521    BM
2  21.743190  0.711865  0.065497  0.182556 -0.237405    BM
3  20.343507  0.835805 -0.218298  0.074244 -0.006637    BM
4  20.212268  0.695531 -0.362522  0.164990  0.177928    BM
<Figure size 450x300 with 0 Axes>
```



In [6]:
```
#Compute eigenvalues and PCA using SVD decomposition
u, s, v = np.linalg.svd(X,full_matrices=True, compute_uv=True)
```

In [7]:
```
print(u)
print(s)
print(v)
```

```
[[-0.03546327  0.01854139 -0.01929457 ... -0.10417738 -0.12005033
  -0.10877857]
 [-0.03917592  0.03498504 -0.0317578  ...  0.02604507  0.05565565
  -0.08676007]
 [-0.04148518  0.01858243 -0.05839645 ...  0.10959718  0.13910646
   0.06993148]
 ...
 [-0.09172846  0.02835099  0.11206833 ...  0.97623376 -0.02674615
  -0.01557316]
 [-0.09345082  0.01675919  0.10163478 ... -0.02629769  0.94805636
  -0.01726308]
 [-0.10061457  0.10813998  0.05423636 ... -0.01521101 -0.01823161
   0.96762135]]
[787.20344447  17.00727798  14.90497111   5.45585597   4.02305552]
[[-0.28660671 -0.23254699 -0.59067402 -0.66913638 -0.25911232]
 [ 0.09628042  0.94586525 -0.26468376 -0.08269154 -0.13846889]
 [ 0.5796977   0.00217365  0.07770962 -0.54889753  0.59717549]
 [-0.72954102  0.18019692  0.05065615 -0.04888869  0.65600519]
 [ 0.20076638 -0.13705902 -0.75659889  0.49167063  0.35598567]]
```

In [8]:
```
s/(n-1)
#eigenvalue is the diagonal entry of (st * s)/(n-1)
```

Out[8]:
```
array([3.9557962 , 0.08546371, 0.07489935, 0.02741636, 0.02021636])
```

In [9]:
```
Xt = np.transpose(X)
B = np.dot(Xt,X)
print(B) #B is the Gram matrix
```

```
[[ 50997.22  41324.26 104906.98 118773.04  46080.15]
 [ 41324.26  33771.67  85049.76  96402.91  37305.  ]
 [104906.98  85049.76 216237.93 244917.66  94861.55]
 [118773.04  96402.91 244917.66 277534.73 107375.01]
 [ 46080.15  37305.    94861.55 107375.01  41705.07]]
```

In [10]:
```
w, v = np.linalg.eig(B) #eigendecompositon of B
print(w)
print(v) # v= D*(D^t)
```

```
[6.19689263e+05 2.89247504e+02 2.22158164e+02 2.97663644e+01
 1.61849757e+01]
[[-0.28660671  0.09628042  0.5796977  -0.72954102  0.20076638]
 [-0.23254699  0.94586525  0.00217365  0.18019692 -0.13705902]
 [-0.59067402 -0.26468376  0.07770962  0.05065615 -0.75659889]
 [-0.66913638 -0.08269154 -0.54889753 -0.04888869  0.49167063]
 [-0.25911232 -0.13846889  0.59717549  0.65600519  0.35598567]]
```

In [11]:
```
#PCA projection is just UD
```

Problem Sheet 1

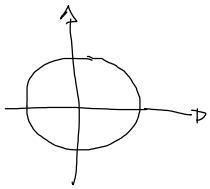
29 January 2021 18:05

Q1. X p-dimensional vector zero mean covariance matrix Σ

a) Larger eigenvalue λ_1 to have multiplicity 1

$\lambda_1 > \lambda_2$

Not identifiable: $p=2$ $X \sim \mathcal{N}(0, I)$



b)

Q2 PCA on centered $(x_1, \dots, x_n)$ $\frac{1}{n}\sum_{i=1}^n x_i = 0$

$S = \frac{1}{n-1}\sum_{i=1}^n x_i x_i^T$ $v_1, \dots, v_p$ the PC.

Denote z_{ij} the projection of x_i onto the jth PC v_j

$(z_{1j}, \dots, z_{nj})$ $z_j = v_j^T x_i$

$$\frac{1}{n}\sum_{i=1}^n z_{ij} = \frac{1}{n}\sum_{i=1}^n v_j^T x_i = \frac{1}{n} v_j^T \underbrace{\left(\sum_{i=1}^n x_i\right)}_{=0} = 0$$

$$\frac{1}{n-1}\sum_{i=1}^n z_{ij}^2 = \frac{1}{n-1}\sum_{i=1}^n \left(v_j^T x_i\right)^2 = \frac{1}{n-1}\sum_{i=1}^n v_j^T x_i x_i^T v_j = v_j^T \left(\frac{1}{n-1}\sum x_i x_i^T\right) v_j$$

$$= v_j^T S v_j = \lambda_j v_j^T v_j = \lambda_j$$

Total variance:

$$\sum_{j=1}^p S_{jj} = \text{trace}(S) \qquad S = V \Lambda V^T \qquad \text{trace}(AB) = \text{trace}(BA)$$

$$= \text{trace}(V \Lambda V^T) = \text{trace}(\Lambda \underbrace{V^T V}_{I}) = \text{trace}(\Lambda) = \sum_{j=1}^p \lambda_j$$

Q3 $h_A(x) = AA^T x$.

(a) $\hat{R}_n(h_A) = \frac{1}{n}\sum_{i=1}^n \| x_i - \underbrace{h_A(x_i)}_{AA^T x_i} \|_2^2$

(b) $= \frac{1}{n}\sum_{i=1}^n (x_i - AA^T x_i)^T (x_i - AA^T x_i)$

$= \frac{1}{n}\sum_{i=1}^n \left[x_i^T x_i - 2 x_i^T AA^T x_i + \underbrace{x_i^T AA^T AA^T x_i}_{I} \right]$ $A^T A = I_{k \times k}$

$= \frac{1}{n}\sum_{i=1}^n \left[\underbrace{x_i^T x_i}_{\mathbb{R}} - x_i^T AA^T x_i \right]$

① $\sum_{i=1}^n x_i^T x_i = \text{trace}\left(\sum_{i=1}^n x_i^T x_i\right) = \sum_{i=1}^n \text{trace}(x_i^T x_i)$ AB BA $n \times t$ $t \times n$ $\text{trace}(AB) = \text{trace}(BA)$

$= \sum_{i=1}^n \text{trace}(x_i x_i^T) = (n-1)\,\text{trace}(S)$

$\left(S = \frac{1}{n-1}\sum_{i=1}^n x_i x_i^T.\right)$

$\sum_{i=1}^n x_i^T AA^T x_i = \sum_{i=1}^n \text{trace}(x_i^T AA^T x_i) = \sum_{i=1}^n \text{trace}(\dots)$

$\sum_{i=1}^{n} x_i^T A^T x_i = \text{trace}(A^T x_i x_i^T A) = \text{trace}(A^T S A)$

$= (n-1)\,\text{trace}(A^T S A)$

② $\sum_{i=1}^{n} x_i^T x_i = \text{trace}(XX^T) = \text{trace}(X^T X) = (n-1)\,\text{trace}(S)$

$X = \begin{pmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{pmatrix}$ $\quad (XX^T)_{ii} = x_i^T x_i$

$XA = \begin{pmatrix} -x_1^T A- \\ \end{pmatrix}$

(c) $\hat{R}_n(h_A) = \frac{n-1}{n}\left[\text{trace}(S) - \underbrace{\text{trace}(A^T S A)}_{\text{maximize}}\right]$

$= \sum_{j=1}^{k} a_j^T S a_j$ $\quad a_j$ column of A.

subject to $a_j^T a_k = 0$ for $j \neq k$

$a_j^T a_j = 1$.

$\max a_1^T S a_1$

s.t. $a_1^T a_1 = 1$.

a_1 eigenvector λ_1

$\vdots$ $\quad\vdots$

a_k eigenvectors λ_k

$\sum_{j=1}^{k} a_j^T S a_j = \sum_{j=1}^{k} \lambda_j = \text{trace}(A^T S A)$

$\text{trace}(S) = \sum_{i=1}^{p} \lambda_i$

$\hat{R}_n(h_A) = \frac{n-1}{n} \sum_{i=k+1}^{p} \lambda_i$

Q4. Dissimilarity matrix A (n by n) $\quad a_{ij} = \|x_i - x_j\|$

n by n matrix B (Gram matrix) $\quad b_{ij} = x_i^T x_j$

a) $a_{ij}^2 = \|x_i - x_j\|^2 = \|x_i\|^2 + \|x_j\|^2 - 2\underbrace{x_i^T x_j}_{b_{ij}}$

(*) $\sum_{j=1}^{n} a_{ij}^2 = n\|x_i\|^2 + \sum_{j=1}^{n}\|x_j\|^2 - 2x_i^T\underbrace{\left(\sum_{j=1}^{n} x_j\right)}_{0}$

$\sum_{ij=1}^{n} a_{ij}^2 = 2n \sum_{k=1}^{n} \|x_k\|^2$

$\|x_i\|^2 = \frac{1}{n}\left(\sum_{j=1}^{n} a_{ij}^2 - \frac{1}{2n}\sum_{ij} a_{ij}^2\right)$

$b_{ij} = \frac{1}{2}\|x_i\|^2 + \frac{1}{2}\|x_j\|^2 - \frac{1}{2}a_{ij}^2$

$b_{ij} = -\frac{1}{2}\left(\frac{1}{n}\sum_{k=1}^{n} a_{ik}^2 - \frac{1}{n}\sum a_{ks}^2\right) + \frac{1}{n}\left(\sum a_{kj}^2 - \frac{1}{n}\sum a_{ks}^2\right) - \frac{1}{2}a_{ij}^2$

$$= \frac{1}{2n}\left(\sum_k a_{kj}^2 + \sum_k a_{ik}^2\right) - \frac{1}{2n^2}\sum_{k,s} a_{ks}^2 - \frac{1}{2}a_{ij}^2$$

J : $J_{ij} = 1$ $\quad \tilde{A}$: $\tilde{A}_{ij} = a_{ij}^2$

$$B = -\frac{1}{2}\left(I - \frac{1}{n}J\right)\tilde{A}\left(I - \frac{1}{n}J\right)$$

b) $X = UDV^T$ the SVD decomposition.

data matrix $n \times p$ $\quad B = XX^T = UDV^TVD^TU^T = U\underbrace{DD^T}_{\tilde{\Lambda}}U^T = U\tilde{\Lambda}U^T$

Eigenvectors : U

Eigenvalues : $\tilde{\Lambda}$

$$Z = XV = UDV^TV = UD$$

Q5 a) $\tilde{W}(C_1, \dots, C_k) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i' \in C_k}\|x_i - x_{i'}\|^2$

Show $\tilde{W}(C_1, \dots, C_k) = \sum_{k=1}^{K}\sum_{i \in C_k}\|x_i - \bar{x}_k\|$

$$\frac{1}{2n_k}\sum_{i,i' \in C_k}\|x_i - x_{i'}\|^2 = \frac{1}{2n_k}\sum_{i,i' \in C_k}\|(x_i - \bar{x}_k) - (x_{i'} - \bar{x}_k)\|^2$$

$$= \frac{1}{2n_k}\sum_{i,i' \in C_k}\left[\|x_i - \bar{x}_k\|^2 + \|x_{i'} - \bar{x}_k\|^2 - 2(x_i - \bar{x}_k)^T(x_{i'} - \bar{x}_k)\right]$$

$$= \frac{1}{2}\sum_{i \in C_k}\|x_i - \bar{x}_k\|^2 + \frac{1}{2}\sum_{i' \in C_k}\|x_{i'} - \bar{x}_k\|^2$$

$$= \sum_{i \in C_k}\|x_i - \bar{x}_k\|^2 \qquad - \frac{2}{2n_k}\Big(\sum_{i \in C_k}(x_i - \bar{x}_k)\Big)^T\Big(\sum_{i' \in C_k}(x_{i'} - \bar{x}_k)\Big)$$

$\Big(\sum_{i' \in C_k}(x_{i'} - \bar{x}_k)\Big) = 0$

b) $(n-1)S = \sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T = \sum_{k=1}^{K}\sum_{i \in C_k}(x_i - \bar{x})(x_i - \bar{x})^T$

$$= \sum_{k=1}^{K}\sum_{i \in C_k}\left((x_i - \bar{x}_k) - (\bar{x} - \bar{x}_k)\right)\left((x_i - \bar{x}_k) - (\bar{x} - \bar{x}_k)\right)^T$$

$$= \sum_k \sum_{i \in C_k}\Big[(x_i - \bar{x}_k)(x_i - \bar{x}_k)^T + (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T\Big] +$$

$$\sum_k \sum_{i \in C_k}\Big[(x_i - \bar{x}_k)(\bar{x}_k - \bar{x})^T + (\bar{x}_k - \bar{x})(x_i - \bar{x}_k)^T\Big]$$

$= 0$

$$= (n-1)W + (n-1)B$$

c) $\tilde{W}(C_1, \dots, C_k) = \sum_{k=1}^{K}\sum_{i \in C_k}\|x_i - \bar{x}_k\|^2 = (n-1)\,\text{trace}(W)$

d) Let $z_i = k$ if $i \in C_k$

For $i = 1, \dots, n$ $\quad \tilde{x}_i = \bar{x}_{z_i}$

$\tilde{T}$ total variance of $(\tilde{x}_1, \dots, \tilde{x}_n)$

$S = W + B \Rightarrow \text{trace}(W) = \text{trace}(S) - \text{trace}(B)$

Minimising $\tilde{W} = \text{trace}(W)$ equivalent to maximising $\text{trace}(B)$

$$\text{trace}(B) = \frac{1}{n-1} \sum_{k=1}^{K} n_k \| \bar{x}_k - \bar{x} \|^2$$

$$\frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i = \frac{1}{n} \sum_{k=1}^{K} n_k \bar{x}_k = \bar{x}$$

Total sample variance of $(\tilde{x}_1, \dots, \tilde{x}_n)$

$$\tilde{T} = \frac{1}{n-1} \sum_{i=1}^{n} \| \tilde{x}_i - \bar{x} \|^2$$

$$= \frac{1}{n-1} \sum_{i=1}^{n} \| \bar{x}_{z_i} - \bar{x} \|^2 = \frac{1}{n-1} \sum_{k=1}^{n} n_k \| \bar{x}_k - \bar{x} \|^2 = \text{trace}(B) \leq \text{trace}(S)$$

$\frac{\text{trace}(B)}{\text{trace}(S)} \leq 1$ proportion of total variance explained by the clustering.

Statistical Machine Learning

Sheet 1 Solutions — HT21

Unsupervised learning

1. Let X be a p-dimensional random vector with zero mean and covariance matrix Σ.

   (a) Under what condition will the first principal component direction be identifiable? (It is not identifiable if there are more than one direction satisfying the defining criterion).

   (b) Supposing it is not identifiable, can you describe the behaviour of the first empirical principal component when computed using a dataset $(x_1, \dots, x_n)$, then on $(x_1, \dots, x_{n+m})$, $m \geq 1$?
   [*hint: what happens when PCA is applied to samples from an isotropic Gaussian?*]

Solution: If the largest eigenvalue λ_1 of Σ has multiplicity 1 (equivalently, that we have strict inequality $\lambda_1 > \lambda_2$), then the first principal component direction is identifiable since its eigenspace is one-dimensional. Otherwise, it is not identifiable, as any of the infinitely many directions within the eigenspace will satisfy the maximum variance criterion.

The computed first PC will not be stable to the addition of new observations; since small perturbations will cause the direction of largest variance to vary significantly.

2. We perform PCA on a centered dataset $(x_1, \dots, x_n)$. Let $S = \frac{1}{n-1} \sum_{i=1}^n x_i x_i^\intercal$ be the sample covariance. Denote the projection of x_i onto the jth (empirical) principal component by z_{ij}.

   (a) Find the sample mean and sample variance of the projections on the jth PC $(z_{1j}, \dots, z_{nj})$

   (b) Show that the total sample variance of $(x_1, \dots, x_n)$ is equal to the sum of the sample variances of the PC projections.

Solution: From $z_{ij} = v_j^\intercal x_i$, $\sum_{i=1}^n z_{ij} = v_j^\intercal \left(\sum_{i=1}^n x_i\right) = 0$, so projections are also centered. The sample variance is then given by

$$\frac{1}{n-1} \sum_{i=1}^n z_{ij}^2 = \frac{1}{n-1} \sum_{i=1}^n v_j^\intercal x_i x_i^\intercal v_j = v_j^\top S v_j = \lambda_j.$$

The total sample variance is $\sum_{j=1}^p S_{jj} = \operatorname{trace}(S)$. But, using the eigendecomposition $S = V \Lambda V^\intercal$, and the property $\operatorname{trace}(AB) = \operatorname{trace}(BA)$, we have $\operatorname{trace}(S) = \operatorname{trace}(V \Lambda V^\top) = \operatorname{trace}(V^\intercal V \Lambda) = \operatorname{trace}(\Lambda) = \sum_{j=1}^p \lambda_j$, which proves the claim.

François Caron: caron@stats.ox.ac.uk

3. Let $(x_1, \ldots, x_n)$ be a centered dataset, where $x_i \in \mathbb{R}^p$. For a p-by-K matrix A with orthonormal columns, that is $A^\intercal A = I_K$, consider the linear encoder and decoder defined by

$$z_i = \mathrm{enc}_A(x_i) = A^\intercal x_i \tag{1}$$

$$\widehat{x}_i = \mathrm{dec}_A(z_i) = Az_i \tag{2}$$

Let $h_A(x_i) = AA^\intercal x_i = \widehat{x}_i$ be the corresponding linear autoencoder.

(a) Give the empirical risk $\widehat{R}_n(h_A)$ of the autoencoder h_A under a squared loss, as a function of $(x_i)_{i=1,\ldots,n}$, A and n.

Solution:

$$\widehat{R}_n(h_A) = \frac{1}{n}\sum_{i=1}^n \|x_i - h_A(x_i)\|^2 = \frac{1}{n}\sum_{i=1}^n \|x_i - AA^\intercal x_i\|^2$$

(b) Show that $\widehat{R}_n(h_A) = \frac{n-1}{n}\left(\mathrm{trace}(S) - \mathrm{trace}(A^\intercal S A)\right)$ where S is the sample covariance matrix.

Solution:

$$\begin{aligned}
\widehat{R}_n(h_A) &= \frac{1}{n}\sum_{i=1}^n \|x_i - AA^\intercal x_i\|^2 \\
&= \frac{1}{n}\sum_{i=1}^n (x_i - AA^\intercal x_i)^\intercal (x_i - AA^\intercal x_i) \\
&= \frac{1}{n}\sum_{i=1}^n x_i^\intercal x_i - 2x_i^\intercal AA^\intercal x_i + x_i^\intercal AA^\intercal AA^\intercal x_i \\
&= \frac{1}{n}\sum_{i=1}^n x_i^\intercal x_i - x_i^\intercal AA^\intercal x_i
\end{aligned}$$

as $A^\intercal A = I_K$. We use the following properties of the trace. If x is a scalar, $\mathrm{trace}(x) = x$, $\mathrm{trace}(A+B) = \mathrm{trace}(A) + \mathrm{trace}(B)$, $\mathrm{trace}(AB) = \mathrm{trace}(BA)$. It follows

$$\begin{aligned}
\sum_{i=1}^n x_i^\intercal x_i = \mathrm{trace}\Big(\sum_{i=1}^n x_i^\intercal x_i\Big) &= \sum_{i=1}^n \mathrm{trace}(x_i^\intercal x_i) = \sum_{i=1}^n \mathrm{trace}(x_i x_i^\intercal) \\
&= \mathrm{trace}\Big(\sum_{i=1}^n x_i x_i^\intercal\Big) = (n-1)\,\mathrm{trace}(S)
\end{aligned}$$

where S is the sample covariance matrix. Similarly,

$$\begin{aligned}
\sum_{i=1}^n x_i^\intercal AA^\intercal x_i = \mathrm{trace}\Big(\sum_{i=1}^n x_i^\intercal AA^\intercal x_i\Big) &= \sum_{i=1}^n \mathrm{trace}(x_i^\intercal AA^\intercal x_i) \\
&= \sum_{i=1}^n \mathrm{trace}(A^\intercal x_i x_i^\intercal A) = (n-1)\,\mathrm{trace}(A^\intercal S A)
\end{aligned}$$

(c) Let $v_1, \ldots, v_p$ be the eigenvectors and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$ be the eigenvalues of S. Show that the p-by-K matrix $V_{1:K} = (v_1, \ldots, v_K)$ is the empirical risk minimiser amongst the class of linear autoencoders h_A, under a squared loss, and give the expression for $\widehat{R}_n(h_{V_{1:K}})$.

Solution: We want to minimise

$$\widehat{R}_n(h_A) = \frac{n-1}{n}\left(\text{trace}(S) - \text{trace}(A^\intercal S A)\right)$$

over the set of orthonormal matrices A, or equivalently maximise

$$\text{trace}(A^\intercal S A) = \sum_{j=1}^{K} a_j^\intercal S a_j \tag{3}$$

where a_j are the columns of A, under the constraints $a_j^\intercal a_k = 0$ for $j \neq k$ and $a_j^\intercal a_j = 1$. Here it is ok to suggest the iterative maximisation procedure seen in the lecture notes, by first finding a_1, then finding a_2 given a_1, etc. Here is a proof to maximise (3) directly (proof non-examinable):

Using the eigendecomposition of S and properties of the trace,

$$\text{trace}(A^\intercal S A) = \text{trace}(A^\intercal V \Lambda V^\intercal A) = \text{trace}(\Lambda V^\intercal A A^\intercal V)$$

Let $C = V^\intercal A A^\intercal V$ (n-by-n matrix). Then, noting that Λ is a diagonal matrix of eigenvalues,

$$\text{trace}(A^\intercal S A) = \text{trace}(\Lambda C) = \sum_{j=1}^{p} \lambda_j C_{jj} = \sum_{j=1}^{p} \lambda_j \|A^\intercal v_j\|^2$$

Note that, for any j,

$$0 \leq C_{jj} = \|A^\intercal v_j\|^2 \leq 1$$

as v_j has unit norm, and A has orthonormal columns (can consider that A is the p-by-K restriction of an orthogonal p-by-p matrix $\widetilde{A}$, then $\|A^\intercal v_j\|^2 \leq \|\widetilde{A}^\intercal v_j\|^2 = 1$). Additionally,

$$\begin{aligned}\sum_{j=1}^{p} C_{jj} &= \text{trace}(V^\intercal A A^\intercal V) = \text{trace}(A^\intercal V V^\intercal A) \\ &= \text{trace}(A^\intercal A) = \text{trace}(A A^\intercal) = \text{trace}(I_K) = K\end{aligned}$$

The maximum value is obtained when $C_{jj} = 1$ for $j = 1, \ldots, K$, and $C_{jj} = 0$ otherwise. This value is achieved for $A = V_{1:K}$ as, for $j = 1, \ldots, K$, $V_{1:K}^\intercal v_j$ is a unit vector with value 1 at location j. We obtain

$$\text{trace}(V_{1:K}^\intercal S V_{1:K}) = \sum_{j=1}^{K} \lambda_j$$

and

$$\widehat{R}_n(h_{V_{1:K}}) = \frac{n-1}{n}\left(\text{trace}(S) - \text{trace}(V_{1:K}^\intercal S V_{1:K})\right) = \frac{n-1}{n}\sum_{j=K+1}^{p}\lambda_j.$$

4. (a) Under the assumption that your data $(x_1, \ldots, x_n)$ are centered, show that you can compute the $n \times n$ Gram matrix B such that $b_{ij} = x_i^\intercal x_j$ using the dissimilarity matrix A where $a_{ij} = \|x_i - x_j\|$.

Solution: We have

$$a_{ij}^2 = \|x_i - x_j\|^2 = \|x_i\|^2 + \|x_j\|^2 - 2x_i^\intercal x_j$$

so

$$\begin{aligned}\sum_{j=1}^n a_{ij}^2 &= n\|x_i\|^2 + \sum_{j=1}^n \|x_j\|^2 - 2\left(\underbrace{\sum_{j=1}^n x_j}_{=0}\right)^\top x_i, \\ \sum_{i,j=1}^n a_{ij}^2 &= 2n\sum_{k=1}^n \|x_k\|^2,\end{aligned}$$

Thus,

$$\|x_i\|^2 = \frac{1}{n}\left(\sum_{j=1}^n a_{ij}^2 - \frac{1}{2n}\sum_{i,j=1}^n a_{ij}^2\right).$$

By inserting into

$$b_{ij} = \frac{1}{2}\|x_i\|^2 + \frac{1}{2}\|x_j\|^2 - \frac{1}{2}a_{ij}^2$$

we obtain

$$\begin{aligned}b_{ij} &= \frac{1}{2n}\left(\sum_{k=1}^n a_{ik}^2 - \frac{1}{2n}\sum_{k,s=1}^n a_{ks}^2\right) + \frac{1}{2n}\left(\sum_{k=1}^n a_{kj}^2 - \frac{1}{2n}\sum_{k,s=1}^n a_{ks}^2\right) - \frac{1}{2}a_{ij}^2 \\ &= \frac{1}{2n}\left(\sum_{k=1}^n a_{kj}^2 + \sum_{k=1}^n a_{ik}^2\right) - \frac{1}{2n^2}\sum_{k,s=1}^n a_{ks}^2 - \frac{1}{2}a_{ij}^2.\end{aligned}$$

Putting together into matrix form, one can check this leads to

$$B = -\frac{1}{2}\left(I - \frac{1}{n}J\right)\widetilde{A}\left(I - \frac{1}{n}J\right)$$

where $\widetilde{A}$ and J are n-by-n matrices defined as $\widetilde{A}_{ij} = a_{ij}^2$ and $J_{ij} = 1$.

(b) Assume that you only observe pairwise euclidian distances a_{ij} between pairs of observations (not the raw data $(x_1, \ldots, x_n)$). Explain how you can obtain the PCA projections. Can you recover the principal components?

Solution: The process of finding a latent representation of the data based on pairwise distances is known as multidimensional scaling. Let $\mathbf{X} = UDV^\intercal$ be the SVD decomposition of the n-by-p data matrix $\mathbf{X}$, where U is an orthogonal n-by-n matrix, D is n-by-p diagonal matrix and V is p-by-p orthogonal matrix.

Note that $\widetilde{\Lambda} = DD^\intercal$ is an n-by-n diagonal matrix, with diagonal entries $\widetilde{\Lambda}_{ii} = D_{ii}^2$ for $i \leq \min(n, p)$ and $\widetilde{\Lambda}_{ii} = 0$ otherwise. There is a one-to-one mapping between $\widetilde{\Lambda}$ and D as $D = \widetilde{\Lambda}^{1/2}C$ where C is the n-by-p matrix with 1 on the diagonal and 0 elsewhere.

The Gram matrix $B = \mathbf{X}\mathbf{X}^\intercal$, which can be obtained from the matrix of pairwise distances A (from part (a)), admits the eigendecomposition

$$\mathbf{X}\mathbf{X}^\intercal = UDV^\intercal VD^\intercal U^\intercal = UDD^\intercal U^\intercal = U\widetilde{\Lambda}U^\intercal.$$

U is therefore the n-by-n matrix of eigenvectors of B, and $\widetilde{\Lambda}$ is the n-by-n diagonal matrix of eigenvalues of B. The projections are obtained as

$$\mathbf{Z} = \mathbf{X}V = UDV^\intercal V = UD.$$

The principal components V cannot be recovered from the pairwise distances.

5. Let $x_1, \ldots, x_n$ be a dataset where $x_i \in \mathbb{R}^p$ and $\Pi = \{C_1, C_2, \ldots, C_K\}$ a partition of $\{1, \ldots, n\}$. For each cluster C_k, denote $n_k = |C_k|$ and define

$$\bar{x}_k = \frac{1}{n_k} \sum_{i \in C_k} x_i \qquad \text{to be the within-cluster sample mean}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^{K} n_k \bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \text{to be the overall sample mean}$$

Let

$$\widetilde{W}(C_1, \ldots, C_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i, i' \in C_k} \|x_i - x_{i'}\|^2$$

be the K-means objective function.

(a) Show that

$$\widetilde{W}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \bar{x}_k\|^2$$

François Caron: caron@stats.ox.ac.uk

Solution:

$$\begin{aligned}
&\frac{1}{2n_k}\sum_{i,i'\in C_k}\|(x_i-\bar{x}_k)-(x_{i'}-\bar{x}_k)\|^2\\
&=\frac{1}{2n_k}\sum_{i,i'\in C_k}\|x_i-\bar{x}_k\|^2+\|x_{i'}-\bar{x}_k\|^2-2(x_i-\bar{x}_k)^\intercal(x_{i'}-\bar{x}_k)\\
&=\frac{1}{2}\sum_{i\in C_k}\|x_i-\bar{x}_k\|^2+\frac{1}{2}\sum_{i'\in C_k}\|x_{i'}-\bar{x}_k\|^2-\frac{1}{n_k}\sum_{i\in C_k}(x_i-\bar{x}_k)^\intercal\left(\sum_{i'\in C_k}(x_{i'}-\bar{x}_k)\right)
\end{aligned}$$

Noting that $\sum_{i'\in C_k}(x_{i'}-\bar{x}_k)=0$, the result follows.

(b) Let

$$S=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(x_i-\bar{x})^\top \qquad \text{be the sample covariance matrix}$$

$$W=\frac{1}{n-1}\sum_{k=1}^{K}\sum_{i\in C_k}(x_i-\bar{x}_k)(x_i-\bar{x}_k)^\top \qquad \text{to be the sample within-cluster covariance matrix}$$

$$B=\frac{1}{n-1}\sum_{k=1}^{K}n_k(\bar{x}_k-\bar{x})(\bar{x}_k-\bar{x})^\top \qquad \text{to be the sample between-cluster covariance matrix}$$

where S, W and B are all $p\times p$ matrices. Show that $S=W+B$

Solution:

$$\begin{aligned}
(n-1)S &= \sum_{k=1}^{K}\sum_{i\in C_k}(x_i-\bar{x}_k+\bar{x}_k-\bar{x})(x_i-\bar{x}_k+\bar{x}_k-\bar{x})^\top\\
&= \sum_{k=1}^{K}\sum_{i\in C_k}\left\{(x_i-\bar{x}_k)(x_i-\bar{x}_k)^\top+(\bar{x}_k-\bar{x})(\bar{x}_k-\bar{x})^\top\right\}+\\
&\quad \sum_{k=1}^{K}\sum_{i\in C_k}\left\{(x_i-\bar{x}_k)(\bar{x}_k-\bar{x})^\top+(\bar{x}_k-\bar{x})(x_i-\bar{x}_k)^\top\right\}\\
&= (n-1)W+(n-1)B+0+0
\end{aligned}$$

since $\sum_{i\in C_k}(x_i-\bar{x}_k)=0$, etc. The sample covariance matrix is the sum of the within and between cluster sample covariance matrices.

(c) Explain how the K-means objective function $\widetilde{W}$ is related to the matrix W.

Solution:

$$\widetilde{W}(C_1,\ldots,C_K)=(n-1)\,\mathrm{trace}(W)$$

(d) Let $z_i=k$ if $i\in C_k$. For $i=1,\ldots,n$, write $\widetilde{x}_i=\bar{x}_{z_i}$. Show that the total sample

variance $\widetilde{T}$ of $\widetilde{x}_1, \ldots, \widetilde{x}_n$ is upper bounded by the total sample variance T of $(x_1, \ldots, x_n)$. Show that minimising $\widetilde{W}$ is equivalent to maximising $\widetilde{T}$.

Solution: Note that $\mathrm{trace}(W) = \mathrm{trace}(S) - \mathrm{trace}(B)$. As S does not depend on the partition, minimising $\mathrm{trace}(W)$ is equivalent to maximising $\mathrm{trace}(B) = \frac{1}{n-1}\sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|^2$. First note that

$$\frac{1}{n}\sum_{i=1}^{n} \widetilde{x}_i = \frac{1}{n}\sum_{k=1}^{K} n_k \bar{x}_k = \bar{x}$$

The total sample variance is

$$\begin{aligned}\widetilde{T} &= \frac{1}{n-1}\sum_{i=1}^{n} \|\widetilde{x}_i - \bar{x}\|^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n} \|\bar{x}_{z_i} - \bar{x}\|^2 = \frac{1}{n-1}\sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|^2 = \mathrm{trace}(B) \leq \mathrm{trace}(S)\end{aligned}$$

Similarly to PCA, $\mathrm{trace}(B)/\mathrm{trace}(S)$ can be interpreted as the proportion of total variance explained by the clustering, and is the quantity we aim at maximising.

6. The following Python code loads the crabs datasets into a data frame $\mathbf{X}$.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from scipy.linalg import eigh # eigendecomposition
from numpy.linalg import svd # SVD

# Load the crabs dataset
url = 'https://vincentarelbundock.github.io/Rdatasets/csv/MASS/crabs.csv'
crabs = pd.read_csv(url)
X = crabs[['FL', 'RW', 'CL', 'CW', 'BD']]
```

Using Python (recommended) or R:

(a) Compute the sample covariance S

(b) Compute the eigenvalues and the principal components using the eigendecomposition of S

(c) Compute the PCA projections, and plot them using a pairplot (function pairplot in seaborn)

(d) Compute the eigenvalues and the principal components using the SVD decomposition of $\mathbf{X}$

(e) Compute the Gram matrix B

(f) Compute the PCA projections from the eigendecomposition of B

Solution: See the Python notebook.

François Caron: caron@stats.ox.ac.uk

7. (Optional) Spike sorting is an important problem in computational neuroscience. Based on recordings of neuronal activity, it aims at detecting spikes, and assigning each spike to the activity of a given neuron. The dataset we will consider is composed of $n = 1000$ spike recordings. For each spike, a set of $p = 96$ features have been extracted using some preprocessing tools. The data[1] are stored in the csv file spike_data.csv. Here is the code to load the data as a panda frame.

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
X = pd.read_csv('spike_data.csv')
```

Using Python (recommended) or R:

(a) Apply PCA to the data X, and compute the projections of the data onto the first two components.

(b) Apply Kmeans with two clusters to the PCA projections, and visualise the obtained partition.

(c) Let $\widehat{\mu}_1 \in \mathbb{R}^2$ and $\widehat{\mu}_2 \in \mathbb{R}^2$ be the estimated cluster means. Map these means into the original data space of dimension 96 using the PCA decoder. Plot the two transformed cluster means.

Solution: See the Python notebook.

[1] This is a subset of the data available from: https://ifcs.boku.ac.at/repository/data/spike_sorting/index.html

François Caron: caron@stats.ox.ac.uk

Statistical Machine Learning

Sheet 2 — HT21

Risk, Bayes prediction rule, Empirical risk, Generative Classifiers

1. Let (X, Y) be input/target random variables taking values in $\mathcal{X} \times \mathcal{Y}$. For a given loss function L, the risk $R(h)$ of a prediction rule h is given by the expected loss

$$R(h) = \mathbb{E}\left[L(Y, h(X))\right].$$

(a) Consider a regression problem ($\mathcal{Y} = \mathbb{R}$) and the squared error loss

$$L(Y, f(X)) = (Y - f(X))^2.$$

(i) Derive the expression of the Bayes prediction rule h^* that minimises $R(h)$.

(ii) Show that the Bayes risk is

$$R(h^*) = \mathbb{E}[\text{var}(Y|X)].$$

(iii) Show that, for a prediction rule h, the excess risk is

$$R(h) - R(h^*) = \mathbb{E}\left[(h(X) - h^*(X))^2\right].$$

(b) Consider a regression problem, with the absolute loss

$$L(Y, f(X)) = |Y - f(X)|.$$

Derive the expression of the Bayes prediction rule h^* that minimises $R(h)$.

Hint: You may use the Leibniz integral rule

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} g(x,t)\,dt\right) = g\big(x, b(x)\big) \cdot \frac{d}{dx}b(x) - g\big(x, a(x)\big) \cdot \frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} g(x,t)\,dt.$$

(c) Consider now a multiclass classification problem ($\mathcal{Y} = \{1, \ldots, K\}$) and the 0-1 loss

$$L(Y, h(X)) = \begin{cases} 0 & \text{if } Y = h(X) \\ 1 & \text{if } Y \neq h(X) \end{cases}$$

Show that

$$h^*(x) = \arg\max_{k \in \mathcal{Y}} \Pr(Y = k | X = x)$$

$$R(h^*) = 1 - \mathbb{E}_X\left[\max_{k \in \mathcal{Y}} \Pr(Y = k|X)\right] \leq \frac{K-1}{K}$$

François Caron: caron@stats.ox.ac.uk

(d) Consider a binary classification problem ($\mathcal{Y} = \{1, -1\}$) with the 0-1 loss. Show that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

$$R(h^*) = \frac{1}{2} - \mathbb{E}\left[|\eta(X) - 1/2|\right]$$

$$R(h) - R(h^*) = \mathbb{E}\left[|2\eta(X) - 1| \mathbb{1}_{h^*(X) \neq h(X)}\right]$$

where $\eta(x) = \Pr(Y = 1 \mid X = x)$.

$L(Y, h(x)) = \begin{cases} 0 & Y = h(x) \\ 1 & Y \neq h(x) \end{cases}$

$h^*(x) = \arg\min_h R(h(x))$

$R(h(x)) = \mathbb{E}[L(Y, h(x))]$

$= \mathbb{P}(Y \neq h(x)) = \int_x \mathbb{P}(Y \neq h(x) \mid X = x) f_X(x) dx$

$\Rightarrow \arg\min \mathbb{P}(Y \neq h(x) \mid X = x) = \arg\max \mathbb{P}(Y = h(x) \mid X = x)$

If $\eta(x) = \mathbb{P}(Y = 1 \mid X = x) \geq \frac{1}{2}$, $h^* = 1$

$\eta(x) < \frac{1}{2}$, $h^* = -1$

$R(h^*) = \mathbb{E}[L(Y, h^*(x))] = 1 \cdot \mathbb{P}(Y \neq h^*(x))$

$= 1 - \mathbb{P}(Y = h^*(x))$

$= 1 - \mathbb{E}_X(Y = h^*(x) \mid X = x)$

$= \int 1 - \mathbb{E}[\eta(x)]$

$h^* = 1$, $\eta(x) \geq \frac{1}{2}$.

$h^* = -1$, $\eta(x) < \frac{1}{2}$

François Caron: caron@stats.ox.ac.uk

2. Consider a binary classification, where the joint distribution of $(X, Y) \in \mathbb{R} \times \{-1, 1\}$ is defined as follows. Y is uniformly distributed on $\{-1, 1\}$, that is $\Pr(Y = 1) = \Pr(Y = -1) = 1/2$, and

$$X|Y = y \sim \begin{cases} \mathcal{N}(\mu, \sigma^2) & \text{if } y = 1 \\ \mathcal{N}(-\mu, \sigma^2) & \text{if } y = -1 \end{cases}$$

for some parameters $\mu \geq 0$, $\sigma > 0$. Denote

$$g_y(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-y\mu)^2}{2\sigma^2}}$$

the conditional probability density function of X given $Y = y$.

(a) Let h^* be the Bayes classifier under a 0-1 loss. Show that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

where

$$\eta(x) = \left(1 + e^{-2x\mu/\sigma^2}\right)^{-1}.$$

(b) Deduce that

$$h^*(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

(c) Show that the Bayes risk is

$$R(h^*) = 1 - \Phi\left(\frac{\mu}{\sigma}\right)$$

where Φ denotes the cumulative distribution function of the standard normal. How does the Bayes risk relate to the between-class and within-class variances of (X, Y)?

(d) Consider the classifier

$$h(x) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}$$

for some $a \neq 0$. Give the expression of its risk $R(h)$, as a function of μ, σ, a and Φ. Check that $R(h) \geq R(h^*)$.

3. Let $d = ((x_1, y_1), \ldots, (x_n, y_n))$ be a realisation from $D = ((X_1, Y_1), \ldots, (X_n, Y_n))$ where (X_i, Y_i) are iid with the same distribution as (X, Y). Let L be a loss function, and h be a fixed prediction rule, that does not depend on d. Assume $R(h) = \mathbb{E}[L(Y, h(X))] < \infty$. Let

$$\widehat{R}_n^{(d)}(h) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$$

be the empirical risk of h, which is a realisation of the random variable (as dependent on the random sample D)

$$\widehat{R}_n^{(D)}(h) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, h(X_i)).$$

Note: You may use any standard limit theorem or inequality for random variables. See Chapter 2 of Part A Probability for a refresher.

(a) Show that $\widehat{R}_n^{(D)}(h)$ is an unbiased estimator of $R(h)$.

(b) Show that $\widehat{R}_n^{(D)}(h)$ is a consistent estimator of $R(h)$.

(c) Assume $\sigma(h)^2 := \mathbb{E}[(L(Y, h(X)) - R(h))^2] < \infty$.

(i) Show that

$$\sqrt{n}(\widehat{R}_n^{(D)}(h) - R(h)) \to \mathcal{N}(0, \sigma(h)^2)$$

as n tends to infinity.

(ii) Show that, for any $\epsilon > 0$

$$\Pr(|\widehat{R}_n^{(D)}(h) - R(h)| > \epsilon) \leq \frac{\sigma(h)^2}{n\epsilon^2}.$$

(iii) If $\mathcal{Y} = \{1, \ldots, K\}$ and L is the $0-1$ loss, show that $\sigma(h) \leq 1$. Deduce a sufficient number n of examples so that

$$\Pr(|\widehat{R}_n^{(D)}(h) - R(h)| > \epsilon) \leq \delta,$$

as a function of $\epsilon > 0$ and $\delta > 0$.

$\leq \frac{\sigma(h)^2}{n\epsilon^2}$

$\leq \frac{1}{n\epsilon^2} \leq \delta$

$\Rightarrow \quad n \geq \frac{1}{\delta\epsilon^2}$

François Caron: caron@stats.ox.ac.uk

4. Consider two univariate normal distributions $\mathcal{N}(\mu, \sigma^2)$ with known parameters $\mu_A = 10$ and $\sigma_A = 5$ for class A and $\mu_B = 20$ and $\sigma_B = 5$ for class B. Suppose class A represents the random score X of a medical test of normal patients and class B represents the score of patients with a certain disease. A patient is 100 times more likely to be healthy than to carry the disease.

   (a) Find the optimal decision rule h^* under the 0-1 loss for allocating a new observation x to either class A or B. Under h^*, what is the probability of misclassifying a healthy patient (A) as having the disease (B)? What is the probability of not detecting the disease (that is misclassifying a type B patient as A)? Given that a patient has the disease, what is the probability of not detecting it?

   (b) Repeat (a) if the cost of a false negative (allocating a sick patient to group A) is $\theta > 1$ times that of a false positive (allocating a healthy person to group B). Describe how the rule changes as θ increases. For which value of θ are 84.1% of all patients with disease correctly classified?

5. Suppose we have a binary classification problem, i.e., $\mathcal{Y} = \{-1, 1\}$, and a 2-dimensional input variable X. Assume that the sample means of the two groups are at $\hat{\mu}_{-1} = (-1, -1)^\top$ and $\hat{\mu}_1 = (1, 1)^\top$ respectively. The estimated prior class probabilities $\hat{\pi}_1$ and $\hat{\pi}_{-1}$ are equal.

   (a) Applying LDA, the covariance matrix is estimated to be, for some value of $0 \leq \rho < 1$,

   $$\hat{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

   Find the decision boundary as a function of ρ.

   (b) Suppose instead that, we model each class with its own covariance matrix. We estimate the covariance matrices for group -1 as

   $$\hat{\Sigma}_{-1} = \begin{pmatrix} 5 & 0 \\ 0 & 1/5 \end{pmatrix},$$

   and for group 1 as

   $$\hat{\Sigma}_1 = \begin{pmatrix} 1/5 & 0 \\ 0 & 5 \end{pmatrix}.$$

   Describe the decision rule and draw a sketch of it in the two-dimensional plane.

François Caron: caron@stats.ox.ac.uk

6. The Wine dataset contains the quantities of $p = 13$ constituents found in $n = 178$ Italian wines. The wines come from the same region, but are derived from $K = 3$ different cultivars. A description of the dataset is given at `https://archive.ics.uci.edu/ml/datasets/Wine`. The goal is to build a classifier for predicting the cultivar of a wine based on its constituents. Here is the code to load the data in Python.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler # to normalise the data
from sklearn.datasets import load_wine
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Load wine data
(X,y) = load_wine(return_X_y = True, as_frame = True)

(n, p) = X.shape # number of examples and dimension of each example
K = y.unique().size # number of classes
print('n=', n, ', p=', p, ', K=', K)
Xy = pd.concat([X, y], axis=1) # X and y in a single data frame
```

   (a) Apply LDA to this dataset. Report the training error for the learned classifier.

   (b) Visualise the coefficients b_k of the decision boundaries.

   (c) Plot the projections of the data in the 2D LDA discriminant space.

François Caron: caron@stats.ox.ac.uk

7. (optional) We consider a dataset of $n = 1797$ 8-by-8 images (hence $p = 64$) of handwritten digits ($K = 10$ classes, labelled 0 to 9). The following Python code loads the dataset, plots some sample images, and splits the dataset into a training set and a test set. The objective is to train a classifier on the training set, and predict the labels of images on the test set.

```
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
# Load data
(X, y) = load_digits(return_X_y=True, as_frame=True, n_class = 10)
# Plot some images in the dataset
for i in range(30):
    plt.subplot(3, 10, i + 1)
    image = X.loc[i,:].to_numpy()
    plt.imshow(image.reshape(8, 8), cmap=plt.cm.gray)
    plt.title(y[i])
    plt.xticks(())
    plt.yticks(())
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

   (a) Train the LDA classifier on the training set. Report the accuracy (proportion of correctly classified images) of the classifier on both the training set and the test set.

   (b) Show the projection of the training data in the 2D LDA discriminant space.

   (c) For comparison, show the projection of the training data onto the first two principal components of PCA

   (d) Train the QDA classifier on the training set, with no regularisation. Report the accuracy of the classifier on the training and test set.

   (e) Retrain the QDA classifier, now using some regularisation. Report the accuracy of the classifier on the training and test set.

   (f) Train the Naive Bayes classifier with Gaussian likelihoods on the training set. Report the accuracy of the classifier on the training and test set.

SB2.2 Statistical Machine Learning Sheet 2. Ruizhen Ma.

1. (a) (i) $R(h) = \mathbb{E}[L(Y, h(X))]$

$= \mathbb{E}[(Y - h(X))^2]$

We have:

$h^*(x) = \arg\min \mathbb{E}[(Y - h(x))^2 \mid X = x]$

$= \arg\min \int \mathbb{E}[(Y - h(x))^2 \mid X = x] f_X(x)\, dx$ $f_X(x)$ is the pdf of X

$\Rightarrow h^*(x) = \arg\min \mathbb{E}[(Y - h(x))^2 \mid X = x]$

$\mathbb{E}[(Y - h(x))^2 \mid X = x] = \mathbb{E}(Y^2 \mid X = x) - 2\mathbb{E}(Y h(x) \mid X = x) + \mathbb{E}(h^2(x) \mid X = x)$

$= \mathbb{E}(Y^2 \mid X = x) - 2h(x)\mathbb{E}(Y \mid X = x) + h^2(x)$

$= \mathrm{Var}(Y \mid X) + \mathbb{E}(Y \mid X)^2 - 2h(x)\mathbb{E}(Y \mid X) + h^2(x)$

$= \mathrm{Var}(Y \mid X) + (\mathbb{E}(Y \mid X) - h(x))^2$ (*)

Hence: $\Rightarrow h^*(x) = \arg\min (\mathbb{E}(Y \mid X) - h(x))^2$

$= \mathbb{E}(Y \mid X)$

(ii) when $h^*(x) = \mathbb{E}(Y \mid X)$, substitute into (*),

$R(h^*) = \mathbb{E}[(Y - \mathbb{E}(Y \mid X))^2]$

$= \mathbb{E}[Y^2 - 2Y\mathbb{E}(Y \mid X) + \mathbb{E}^2(Y \mid X)]$

Since $\mathbb{E}(Y^2) = \mathbb{E}[\mathbb{E}(Y^2 \mid X)]$

$\mathbb{E}[\mathbb{E}^2(Y \mid X) - 2Y\mathbb{E}(Y \mid X)] = \mathbb{E}[-\mathbb{E}^2(Y \mid X)]$

Hence $R(h^*) = \mathbb{E}[\mathbb{E}(Y^2 \mid X) - \mathbb{E}^2(Y \mid X)]$

$= \mathbb{E}[\mathrm{Var}(Y \mid X)]$

(iii) $R(h) - R(h^*)$

$= \mathbb{E}[(Y - h(x))^2 - (Y - h^*(x))^2]$

$= \mathbb{E}[-2Yh(x) + h^2(x) + 2Yh^*(x) - h^{*2}(x)]$

Since $\mathbb{E}[Yh(x)] = \mathbb{E}[\mathbb{E}(Y \mid X) \cdot h(x)] = \mathbb{E}[h^*(x)h(x)]$

$R(h) - R(h^*) = \mathbb{E}[-2h^*(x)h(x) + h^2(x) + 2h^*(x) \cdot h^*(x) - h^{*2}(x)]$

$= \mathbb{E}[(h(x) - h^*(x))^2]$ □.

$$\mathbb{E}[|Y-h(x)| \mid X=x] = \int_{-\infty}^{h(x)} h(x)-y \, f_{Y|X=x}(y)\,dy + \int_{h(x)}^{\infty} (y-h(x)) f_{Y|X=x}(y)\,dy$$

(b) $L(Y, f(x)) = |Y - f(x)|$

$$h^* = \arg\min \mathbb{E}|Y-h(x)|$$

$$\Rightarrow h^* = \arg\max \mathbb{E}(|Y-h(x)| \mid X=x)$$

$$\mathbb{E}(Y-h(x) \mid X=x) = \int_{h(x)}^{\infty} |y-h(x)| f_{Y|X}(y)\,dy$$

change variable ??

$$= \int_{-\infty}^{h(x)} (h(x)-y) k(y)\,dy + \int_{h(x)}^{\infty} (y-h(x)) k(y)\,dy \quad (*)$$

Therefore by Leibniz integral rule:

$$(*)' : \frac{d}{dh(x)} \mathbb{E}(Y-h(x) \mid X=x) = 0 + \int_{-\infty}^{h(x)} k(y)\,dy + 0 + \int_{h(x)}^{\infty} -k(y)\,dy$$

$$= \mathbb{P}(Y \le h(x) \mid X=x) - \mathbb{P}(Y > h(x) \mid X=x)$$

Since we want to set $(*) = 0$.

$$\Rightarrow h(x) = \text{median of } (Y \mid X=x)$$

(c) $L(Y, h(x)) = \begin{cases} 0 & Y = h(x) \\ 1 & Y \ne h(x) \end{cases}$

$$h^* = \arg\min \mathbb{E}[L(Y, h(x))] = \arg\min_{y^* \in \mathcal{Y}} \mathbb{E}[L(Y, y^*) \mid X=x].$$

$$= \arg\min_{y^* \in \mathcal{Y}} \mathbb{P}(Y \ne y^* \mid X=x)$$

$$= \arg\min_{y^* \in \mathcal{Y}} \{1 - \mathbb{P}(Y = y^* \mid X=x)\}$$

$$= \arg\max_{y^* \in \mathcal{Y}} \mathbb{P}(Y = y^* \mid X=x)$$

$$R(h^*) = \mathbb{E}[L(Y, h^*(x))] = 1 - \mathbb{P}(Y = h^*(x))$$

$$= 1 - \mathbb{E}_X [\max_{y^* \in \mathcal{Y}} \mathbb{P}(Y=y \mid X=x)] \quad (*)$$

$\mathcal{Y} = \{1, \dots, K\}$, $\max_{k \in \mathcal{Y}} \mathbb{P}(Y=k \mid X=x) \ge \frac{1}{K}$, hence $(*) \le \frac{K-1}{K}$.

(b) $L(Y, h(x)) = |Y - h(x)|$

$$\mathbb{E}[|Y-h(x)| \mid X=x] = \int_{-\infty}^{h(x)} (h(x)-y) f_{Y|X=x}(y)\,dy + \int_{h(x)}^{\infty} (y-h(x)) f_{Y|X=x}(y)\,dy$$

$$\frac{d}{dh(x)} \downarrow = \int_{-\infty}^{h(x)} f_{Y|X=x}(y)\,dy + \int_{h(x)}^{\infty} -f_{Y|X=x}(y)\,dy$$

$$= P(Y < h(x) \mid X=x) - P(Y > h(x) \mid X=x) = 0$$

$h^*(x)$: median of $Y|X=x$.

(d) $\mathcal{Y} = \{-1, 1\}$ $\quad L(Y, h(X)) = \begin{cases} 0 & Y = h(X) \\ 1 & Y \neq h(X) \end{cases}$

Proof: $h^* = \arg\min_{y^* \in \mathcal{Y}} R(h)$

$= \arg\min_{y^* \in \mathcal{Y}} \mathbb{E}[L(Y, y^*) \mid X = x]$

$= \arg\min_{y^* \in \mathcal{Y}} \mathbb{P}(Y \neq y^* \mid X = x)$

If $\mathbb{P}(Y = 1 \mid X = x) \geq \frac{1}{2}$, $\quad h^* = 1$

If $\mathbb{P}(Y = 1 \mid X = x) < \frac{1}{2}$, $\quad h^* = -1$

Now $R(h^*) = \mathbb{E}[L(Y, h^*(X))]$ $\quad \mathbb{E}[L(Y, h(X))] = \mathbb{E}_X[\mathbb{P}(Y \neq h(X) \mid X = x)]$

$= 1 - \mathbb{P}(Y = h^*(X))$ $\quad = 1 - \mathbb{P}(Y = h(X) \mid X = x)$

$= 1 - \mathbb{E}_X(\mathbb{P}(Y = h^*(X) \mid X = x))$

$= \begin{cases} 1 - \mathbb{E}[\eta(X)] & h^* = 1;\ \eta(X) \geq \frac{1}{2} \\ 1 - \mathbb{E}[1 - \eta(X)] & h^* = -1;\ \eta(X) < \frac{1}{2} \end{cases}$

$= \frac{1}{2} - \mathbb{E}[|\eta(X) - \frac{1}{2}|]$ $\quad \square$.

$R(h) - R(h^*) = \mathbb{E}[L(Y, h(X))] - \mathbb{E}[L(Y, h^*(X))]$

$= 1 - \mathbb{P}(Y = h(X)) - \{1 - \mathbb{P}(Y = h^*(X))\}$ $(*)$

if $h(X) = h^*(X)$ $\quad (*) = 0$

if $h(X) \neq h^*(X)$ $\quad (*) = 1 - \mathbb{E}_X(\mathbb{P}(Y = h(X) \mid X = x)) - (\frac{1}{2} - \mathbb{E}[|\eta(X) - \frac{1}{2}|])$

$h(X)$ is binary

so $\quad = 1 - \mathbb{E}_X(\mathbb{P}(Y \neq h^*(X) \mid X = x)) - (\frac{1}{2} - \mathbb{E}[|\eta(X) - \frac{1}{2}|])$

$= \mathbb{E}_X(\mathbb{P}(Y = h^*(X) \mid X = x)) - (\frac{1}{2} - \mathbb{E}[\eta(X) - \frac{1}{2}])$

$= \frac{1}{2} + \mathbb{E}[|\eta(X) - \frac{1}{2}|] - \frac{1}{2} + \mathbb{E}[\eta(X) - \frac{1}{2}]$

$= \mathbb{E}[|2\eta(X) - 1|]$

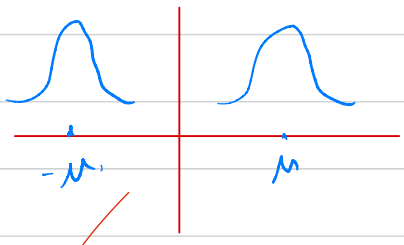
Hence, $R(h) - R(h^*) = \mathbb{E}[|2\eta(X) - 1| \mathbb{1}_{h^*(X) \neq h(X)}]$.

Good work!

2.

(a) $X|Y=y \sim \begin{cases} N(\mu,\sigma^2) & \text{if } y=1 \\ N(-\mu,\sigma^2) & \text{if } y=-1 \end{cases}$

$Y = \{-1, 1\}$



0-1 loss: $L(Y, h(X)) = \begin{cases} 0 & Y = h(X) \\ 1 & Y \neq h(X) \end{cases}$

$$g_y(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(x-y\mu)^2}{2\sigma^2}\right]$$

$$P(Y=k \mid X=x) = \frac{\frac{1}{2} g_k(x)}{\sum_{i=1}^{2} g_i(x) \cdot \pi_i} = \frac{g_k(x)}{\sum_{i=1}^{2} g_i(x)}$$

$$h^*(x) = \underset{y \in Y}{\operatorname{argmax}}\, P(Y=y \mid X=x) = \underset{k \in Y}{\operatorname{argmax}}\, g_y(x)$$

$$= \underset{y \in Y}{\operatorname{argmax}} \exp\left[-\frac{(x-y\mu)^2}{2\sigma^2}\right]$$

$y\mu = \begin{cases} \mu \\ -\mu \end{cases}$

$$= \underset{y \in Y}{\operatorname{argmax}} \exp\left[-\frac{x^2 - 2xy\mu + (y\mu)^2}{2\sigma^2}\right]$$

$$= \underset{y \in Y}{\operatorname{argmax}} \exp\left[\frac{xy\mu}{\sigma^2}\right]$$

If $\eta(x) = (1 + e^{-2x\mu/\sigma^2})^{-1} \geq \frac{1}{2}$

$\Rightarrow (1 + e^{-2x\mu/\sigma^2}) \leq 2$

$\Rightarrow e^{-2x\mu/\sigma^2} \leq 1$

$\Rightarrow x \leq 0$

$(\mu > 0, \sigma^2 > 0)$

Then $\underset{k \in Y}{\operatorname{argmax}} \exp\left[\frac{xy\mu}{\sigma^2}\right]$ is when $y = 1$, hence $h^*(x) = 1$

Similarly when $\eta(x) > \frac{1}{2}$, this implies $x > 0$

Therefore $h^*(x) = -1$

(b) $h^*(x) = \arg\max_{k \in Y} \exp\left[\frac{x y \mu}{\sigma^2}\right]$

if $x \geq 0$, we seek to maximize $y\mu$ $\Rightarrow$ $h^*(x) = 1$

if $x < 0$, we seek to maximize $|x(y\mu)|$ $\Rightarrow$ $h^*(x) = -1$

(c) Bayes risk:

$$R(h^*) = \mathbb{E}[L(Y, h^*(x))]$$
$$= 1 - \mathbb{P}(Y = h^*(x))$$
$$= 1 - \mathbb{E}(\mathbb{P}(Y = h^*(x) \mid X))$$
$$= 1 - \mathbb{E}\left[\frac{\mathbb{P}(X=x \mid Y=h^*(x))\mathbb{P}(Y=h^*(x))}{\mathbb{P}(X=x)}\right]$$
$$=$$

$$R(h^*) = \mathbb{P}(Y \neq h^*(x))$$
$$= \mathbb{P}(Y=1, X<0) + \mathbb{P}(Y=-1, X \geq 0)$$
$$= \mathbb{P}(Y=1)\mathbb{P}(X<0 \mid Y=1) + \mathbb{P}(Y=-1)\mathbb{P}(X \geq 0 \mid Y=-1)$$

(d)
$$= \frac{1}{2}\left(\int_{-\infty}^{0} g_1(x)\,dx + \int_{0}^{\infty} g_{-1}(x)\,dx\right)$$
$$= \frac{1}{2}\left[\Phi\left(\frac{0-\mu}{\sigma}\right) + 1 - \Phi\left(\frac{0+\mu}{\sigma}\right)\right]$$
$$= 1 - \Phi\left(\frac{\mu}{\sigma}\right) = 1 - \Phi\left(\sqrt{\frac{\mu^2}{\sigma^2}}\right)$$

Between class variance is $\mathrm{Var}(\mathbb{E}(x \mid Y)) = \mu^2$

within class variance is $\mathbb{E}(\mathrm{Var}(x \mid Y)) = \sigma^2$.

$\frac{\mu^2}{\sigma^2}$ is a measure of the seperablity of the class.

(d) d) $h(x) = \begin{cases} 1 & \text{if } x \geq a \\ -1 & \text{if } x < a \end{cases}$

$a \neq 0$

$$R(h) = \Pr(Y \neq h(x)) = \Pr(Y=1, h(x)=-1) + \Pr(Y=-1, h(x)=1)$$
$$= \frac{1}{2}\left(\int_{-\infty}^{a} g_1(x)\,dx + \int_{a}^{\infty} g_{-1}(x)\,dx\right) = 1 - \frac{1}{2}\left(\Phi\left(\frac{\mu-a}{\sigma}\right) + \Phi\left(\frac{\mu+a}{\sigma}\right)\right)$$

Differentiate wrt a you can show that the minimum is at $a = 0$

$= 0$ $R(h) \geq R(h^*)$

3.

(a) $\hat{R}_n^{(D)}(h) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, h(X_i))$, $R(h) = \mathbb{E}[L(Y, h(X))]$

Since $\mathbb{E}(\hat{R}_n^{(D)}(h)) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} L(Y_i, h(X_i))\right)$

$= \frac{1}{n} \cdot n \cdot \mathbb{E}[L(Y, h(X))]$

$= R(h)$ □

(b) To show consistent:

Proof: to show $\lim_{n\to\infty} \mathbb{P}(|\hat{R}_n^{(D)}(h) - R(h)| \geq \varepsilon) = 0 \quad \forall \varepsilon > 0$

Since $\mathbb{P}((\hat{R}_n^{(D)}(h) - R(h))^2 \geq \varepsilon^2) \leq \frac{\mathbb{E}[\hat{R}_n^{(D)}(h) - R(h)]^2}{\varepsilon^2}$ by Chebyshev's Inequality.

Since $\mathbb{E}[\hat{R}_n^{(D)}(h) - R(h)]^2 \to 0$ as $n \to \infty$

$\Rightarrow \lim_{n\to\infty} \mathbb{P}(|\hat{R}_n^{(D)}(h) - R(h)| \geq \varepsilon) = 0 \quad \forall \varepsilon > 0$ □

Can use SLLN.

(c) Now assume $\sigma(h)^2 = \mathbb{E}[(L(Y, h(X)) - R(h))^2] < \infty$

(i) Since: $\hat{R}_n^{(D)}(h) - R(h)$

$= \frac{1}{n}\sum_{i=1}^{n} L(Y_i, h(X_i)) - R(h)$

$= \frac{1}{n}\sum_{i=1}^{n} [L(Y_i, h(X_i)) - R(h)]$

Let $Z_i = L(Y_i, h(X_i)) - R(h)$, then $\hat{R}_n^{(D)}(h) - R(h) = \frac{1}{n}\sum_{i=1}^{n} Z_i$

$\mathbb{E}(Z_i) = 0$

$\mathrm{Var}(Z_i) = \mathbb{E}(Z_i^2) - (\mathbb{E}(Z_i))^2$

$= \sigma(h)^2$

Therefore by CLT, $\hat{R}_n^{(D)}(h) - R(h) = \frac{1}{n}\sum_{i=1}^{n} Z_i \sim N(0, \sigma(h)^2/n)$ as $n \to \infty$

$\Rightarrow \sqrt{n}(\hat{R}_n^{(D)}(h) - R(h)) \sim N(0, \sigma(h)^2)$ as $n \to \infty$

(ii) $\mathbb{P}(|\hat{R}_n^{(D)}(h) - R(h)| > \varepsilon) = \mathbb{P}((\hat{R}_n^{(D)}(h) - R(h))^2 > \varepsilon^2) \leq \frac{\mathbb{E}[(\hat{R}_n^{(D)}(h) - R(h))^2]}{\varepsilon^2}$ by Chebyshev's Inequality

Since $\mathbb{E}[(\hat{R}_n^{(D)}(h) - R(h))^2] = \sigma(h)^2/n$

$\Rightarrow \mathbb{P}(|\hat{R}_n^{(D)}(h) - R(h)| > \varepsilon) \leq \frac{\sigma(h)^2/n}{\varepsilon^2} = \frac{\sigma(h)^2}{n\varepsilon^2}$.

(iii) $\mathcal{Y} = \{1, \dots, k\}$ $L(Y, h(X)) = \begin{cases} 0 & Y = h(X) \\ 1 & Y \neq h(X) \end{cases}$ $R(h) = \mathbb{E}(L(Y, h(X)))$ $= \mathbb{P}(Y \neq h(X))$

$\sigma(h)^2 = \mathbb{E}[(L(Y, h(X)) - R(h))^2] \leq 1$ since $|L(Y, h(X)) - R(h)| \leq 1$.

$\Rightarrow \sigma(h) \leq 1$.

Since: $\mathbb{P}(|\hat{R}^{(D)}(h) - R(h)| > \varepsilon) \leq \frac{\sigma(h)^2}{n\varepsilon^2} \leq \frac{1}{n\varepsilon^2}$

$\frac{1}{n\varepsilon^2} \leq \delta$ iff: $n \geq \frac{1}{\delta\varepsilon^2}$

$n \geq \frac{1}{\delta\varepsilon^2}$

4.

(a) $h^* = \arg\max_{k \in \{A,B\}} \mathbb{P}(Y=k \mid X=x) = \arg\max_{k \in \{A,B\}} \pi_k g_k(x)$

$\eta(x) = f_A(x) - f_B(x)$

$= \pi_A \frac{1}{\sqrt{2\pi}\sigma_A} \exp\left(-\frac{(x-\mu_A)^2}{2\sigma_A^2}\right) - \pi_B \frac{1}{\sqrt{2\pi}\sigma_B} \exp\left(-\frac{(x-\mu_B)^2}{2\sigma_B^2}\right) = 0$

$\mu_A = 10, \mu_B = 20, \sigma_A = \sigma_B = 5$ $\quad \pi_A/\pi_B = 100$

$\Rightarrow \quad \eta(x) = -2\sigma_A^2 \log(\pi_A/\pi_B) + (x-\mu_A)^2 - (x-\mu_B)^2 = 0$

$\eta(x) = -2x(\mu_A - \mu_B) + (\mu_A^2 - \mu_B^2) - 2\sigma_A^2 \log(\pi_A/\pi_B) = 0$

$\Rightarrow \quad 20x - 300 - 25 \cdot \ln 100$

$\Leftrightarrow \quad x = \frac{25 \ln 100 + 300}{20} \approx 26.51$

Therefore any patient with score > 26.51 is classified as having the disease.

Under h^*:

$\mathbb{P}(\text{classify } B \mid A) = \mathbb{P}(x > 26.51 \mid A)$

$= \mathbb{P}(x > 26.5 \mid x \sim N(10, 5^2))$

$= \mathbb{P}(z > 0.66)$

$= 0.2546$

$\mathbb{P}(\text{classify } A \mid B) = \mathbb{P}(x < 26.5 \mid B)$

$= \mathbb{P}(x < 26.5 \mid x \sim N(20, 5^2))$

$= \mathbb{P}(z < 0.26)$

$= 0.6026$

$\mathbb{P}(\text{not detecting} \mid \text{has disease}) = \mathbb{P}(x < 26.5 \mid B) = 0.6026$ ✗

$P(Y=A, h^*(x)=B) = P_r(Y=A, X > 26.51)$

$= \pi_A \left(1 - \Phi\left(\frac{26.51 - \mu_A}{\sigma_A}\right)\right) = 0.00047$

$P_r(Y=B, h^*(x)=A) = \pi_B \left(\Phi\left(\frac{26.51 - \mu_B}{\sigma_B}\right)\right) \simeq 0.0089$

$P_r(h^*(x)=A \mid Y=B) = \Phi\left(\frac{26.51 - \mu_B}{\sigma_B}\right) \simeq 0.9035$

(b) $R(h) = \mathbb{E}[L(Y, h(x))]$ $\quad Y = \{A, B\}$

Suppose $h^* = \arg\min R(h)$

$= \arg\min_{y^* \in Y} \mathbb{E}[L(Y, y^* \mid X=x]$

chose $y^* = A$ $\quad L(A,A)P(Y=A|X=x) + L(B,A)P(Y=B|X=x) = 0 + \theta P(Y=B|X=x)$

$y^* = B$ $\quad L(A,B)P(Y=A|X=x) + L(B,B)P(Y=B|X=x) = P(Y=A|X=x) + 0$

So we only chose $y^* = A$ iff $\theta P(Y=B|X=x) \le P(Y=A|X=x)$

$\Rightarrow \theta \pi_B \frac{1}{\sqrt{2\pi}\sigma_B} \exp\left[-\frac{(x-\mu_B)^2}{2\sigma_B^2}\right] \le \pi_A \frac{1}{\sqrt{2\pi}\sigma_A} \exp\left[-\frac{(x-\mu_A)^2}{2\sigma_A^2}\right]$

$\Rightarrow 2\sigma_A^2 \log\left(\frac{\theta\pi_B}{\pi_A}\right) - (x-\mu_B)^2 \le (x-\mu_A)^2$

From (a), the descriminat boundary:

$-2x(\mu_A - \mu_B) + (\mu_A^2 - \mu_B^2) - 2\sigma_A^2 \log\left(\frac{\pi_A}{\theta\pi_B}\right) = 0$

$20x - 300 + 50 \ln\left(\frac{100}{\theta}\right) = 0 \Rightarrow x = \frac{50\ln\left(\frac{100}{\theta}\right) + 300}{20}$

Therefore as θ increases, x with smaller score is more likely to be classified as having the disease.

84.1% correctly classified $\Rightarrow$ $x = \Phi^{-1}(15.9\%) \times 5 + 20$

$= 14.82$

$14.82 = \frac{50\ln\left(\frac{100}{\theta}\right) + 300}{20}$

$\Rightarrow \theta = 107.47$.

3.

(a) $y = \{-1, 1\}$

The decision boundary: $\hat{h}^{(d)}(x) = \arg\max_k \log(\hat{\pi}_k) - \frac{1}{2}(x-\hat{\mu}_k)^T \hat{\Sigma}^{-1}(x-\hat{\mu}_k)$

$$= \arg\max_{k \in Y} \underbrace{\log(\hat{\pi}_k) - \frac{1}{2}\hat{\mu}_k^T \hat{\Sigma}^{-1}\hat{\mu}_k}_{a_k} + \underbrace{\hat{\mu}_k^T \hat{\Sigma}^{-1}}_{b_k^T} x$$

Descriminant boundary: $a_1 + b_1^T x - (a_{-1} + b_{-1}^T x) = 0$

$$(a_1 - a_{-1}) = \underbrace{\log(\hat{\pi}_1/\hat{\pi}_{-1})}_{0} - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1}\hat{\mu}_1 + \frac{1}{2}\hat{\mu}_{-1}^T \hat{\Sigma}^{-1}\hat{\mu}_{-1}$$

$$\hat{\mu}_{-1}^T \hat{\Sigma}^{-1}\mu_{-1} = (1, 1)\begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \times \frac{1}{1-\rho^2}$$

$$\hat{\mu}_1^T \hat{\Sigma}^{-1}\mu_1 = (-1, -1)\begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}\begin{pmatrix} -1 \\ -1 \end{pmatrix} \times \frac{1}{1-\rho^2}$$

$\Rightarrow a_* = (a_1 - a_{-1}) = 0$

$$b_* = b_1 - b_{-1} = \hat{\mu}_1^T \hat{\Sigma}^{-1} - \hat{\mu}_{-1}^T \hat{\Sigma}^{-1}$$

$$= \left[(1, 1)\begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix} - (-1\ -1)\begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}\right] \times \frac{1}{1-\rho^2}$$

$$= \left(\frac{2}{1+\rho}, \frac{2}{1+\rho}\right)$$

Decision boundary: $k = 1$ is chosen iff $a_* + b_*^T x > 0$

$$\Leftrightarrow \left(\frac{2}{1+\rho}, \frac{2}{1+\rho}\right)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} > 0$$

$$\Leftrightarrow x_1 + x_2 > 0$$

(b) $\hat{\Sigma}_{-1} = \begin{pmatrix} 5 & 0 \\ 0 & 1/5 \end{pmatrix}$ $\hat{\Sigma}_{1} = \begin{pmatrix} 1/5 & 0 \\ 0 & 5 \end{pmatrix}$ $Y = \{-1, 1\}$

$$\hat{h}^{(d)}(x) = \underset{k \in Y}{\arg\max} \underbrace{\log(\hat{\pi}_k) - \tfrac{1}{2}\hat{\mu}_k^T \hat{\Sigma}_k^{-1} \hat{\mu}_k}_{\hat{a}_k} + \underbrace{\hat{\mu}_k^T \hat{\Sigma}^{-1}}_{\hat{b}_k^T} x \underbrace{- \tfrac{1}{2} x^T \hat{\Sigma}_k^{-1} x}_{x^T \hat{C}_k x}$$

$a^* = \hat{a}_1 - \hat{a}_{-1} = 0$

$b^* = \hat{b}_1^T - \hat{b}_{-1}^T = (1,1)\begin{pmatrix} 5 & 0 \\ 0 & 1/5 \end{pmatrix} - (-1,-1)\begin{pmatrix} 1/5 & 0 \\ 0 & 5 \end{pmatrix}$

$= (5 \quad 1/5) - (-1/5 \quad -5)$

$= (26/5 \quad 26/5)$

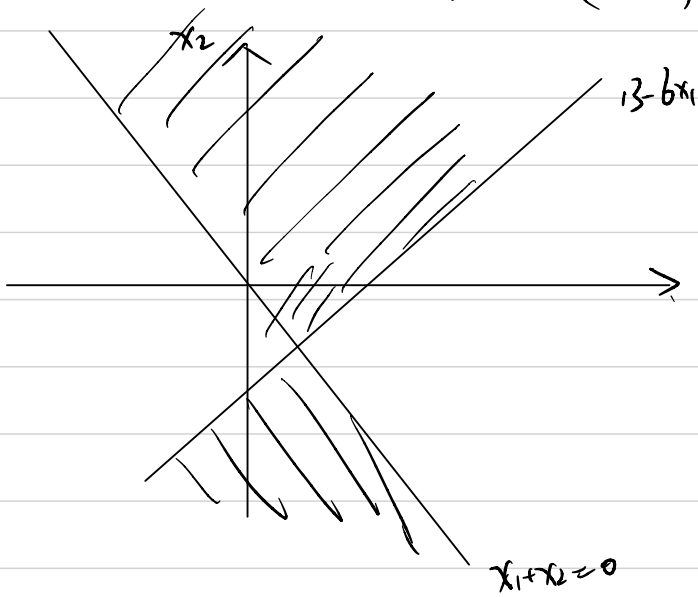
$C^* = \hat{C}_1 - \hat{C}_{-1} = \frac{1}{2}\begin{pmatrix} 5 & 0 \\ 0 & 1/5 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 1/5 & 0 \\ 0 & 5 \end{pmatrix} = \begin{pmatrix} 12/5 & 0 \\ 0 & -12/5 \end{pmatrix}$

Decision boundary: $k=1$ if $\hat{a}_k + \hat{b}_k^T x - x^T \hat{C}_k x > 0$

$\Rightarrow \frac{26}{5}(x_1 + x_2) - \frac{12}{5}(x_1^2 - x_2^2) > 0$

$\Rightarrow 13(x_1 + x_2) - 6(x_1^2 - x_2^2) > 0$

$\Rightarrow (x_1 + x_2)(13 - 6x_1 + 6x_2) > 0$



$13 - 6x_1 + 6x_2 = 0$ $x_1 + x_2$, $13 - 6x_1 + 6x_2$ must be of equal sign.

Statistical Machine Learning

Sheet 2 Solutions — HT21

Risk, Bayes prediction rule, Empirical risk, Generative Classifiers

1. Let (X, Y) be input/target random variables taking values in $\mathcal{X} \times \mathcal{Y}$. For a given loss function L, the risk $R(h)$ of a prediction rule h is given by the expected loss

$$R(h) = \mathbb{E}\left[L(Y, h(X))\right].$$

(a) Consider a regression problem ($\mathcal{Y} = \mathbb{R}$) and the squared error loss

$$L(Y, h(X)) = (Y - h(X))^2.$$

(i) Derive the expression of the Bayes prediction rule h^* that minimises $R(h)$.

(ii) Show that the Bayes risk is

$$R(h^*) = \mathbb{E}[\text{var}(Y|X)].$$

(iii) Show that, for a prediction rule h, the excess risk is

$$R(h) - R(h^*) = \mathbb{E}\left[(h(X) - h^*(X))^2\right].$$

Solution: We have

$$\begin{aligned} R(h) &= \mathbb{E}\left[(Y - h(X))^2\right] \\ &= \int_{\mathcal{X}} \mathbb{E}\left[(Y - h(X))^2 \middle| X = x\right] f_X(x)\, dx, \end{aligned}$$

where f_X is pdf/pmf of X. Thus, it suffices to minimise, for every x,

$$\begin{aligned} & \mathbb{E}\left[(Y - h(X))^2 \middle| X = x\right] \\ = \; & \mathbb{E}\left[Y^2 \middle| X = x\right] - 2h(x)\,\mathbb{E}\left[Y \middle| X = x\right] + h(x)^2 \\ = \; & \text{var}\left[Y|X = x\right] + \left(\mathbb{E}\left[Y \middle| X = x\right] - h(x)\right)^2. \end{aligned}$$

This is clearly minimized by the conditional mean:

$$h^*(x) = \mathbb{E}\left[Y \middle| X = x\right].$$

The associated risk is

$$R(h^*) = \mathbb{E}[\text{var}(Y|X)] = \int_{\mathcal{X}} \text{var}(Y|X = x) f_X(x) dx$$

We have

$$\begin{aligned} R(h) &= \mathbb{E}\left[\text{var}(Y|X) + \left(\mathbb{E}\left[Y \middle| X\right] - h(X)\right)^2\right] \\ &= R(h^*) + \mathbb{E}[(h(X) - h^*(X))^2]. \end{aligned}$$

(b) Consider a regression problem, with the absolute loss

$$L(Y, h(X)) = |Y - h(X)|.$$

Derive the expression of the Bayes prediction rule h^* that minimises $R(h)$.

Hint: You may use the Leibniz integral rule

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} g(x,t)\,dt\right) = g\big(x,b(x)\big)\cdot\frac{d}{dx}b(x) - g\big(x,a(x)\big)\cdot\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{\partial}{\partial x}g(x,t)\,dt.$$

Solution:

As before, we want to find $h(x)$ to minimise

$$\begin{aligned}\mathbb{E}\left[|Y-h(x)|\big|X=x\right] &= \int_{-\infty}^{\infty}|y-h(x)|\,f_Y(y)dy \\ &= \int_{-\infty}^{h(x)}(h(x)-y)\,f_Y(y)dy + \int_{h(x)}^{\infty}(y-h(x))\,f_Y(y)dy\end{aligned}$$

where f_Y is the pdf of Y. We differentiate these integrals with respect to $h(x)$ using the Leibniz integral rule:

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} g(x,t)\,dt\right) = g\big(x,b(x)\big)\cdot\frac{d}{dx}b(x) - g\big(x,a(x)\big)\cdot\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{\partial}{\partial x}g(x,t)\,dt,$$

which gives

$$\begin{aligned}\frac{d}{dh(x)}\mathbb{E}\left[|Y-h(x)|\big|X=x\right] &= \int_{-\infty}^{h(x)} f_Y(y)dy + \int_{h(x)}^{\infty} -f_Y(y)dy \\ &= \Pr(Y<h(x)|X=x) - P(Y>h(x)|X=x).\end{aligned}$$

Setting this to 0, we get the minimum for $\Pr(Y<h(x)|X=x) = \Pr(Y>h(x)|X=x) = 1/2$, i.e. at the conditional median $h^*(x) = \text{med}\left[Y\middle|\,X=x\right]$.

(c) Consider now a multiclass classification problem ($\mathcal{Y} = \{1,\dots,K\}$) and the 0-1 loss

$$L(Y, h(X)) = \begin{cases} 0 & \text{if } Y = h(X) \\ 1 & \text{if } Y \neq h(X)\end{cases}$$

Show that

$$\begin{aligned}h^*(x) &= \arg\max_{k\in\mathcal{Y}} \Pr(Y=k|X=x) \\ R(h^*) &= 1 - \mathbb{E}_X\left[\max_{k\in\mathcal{Y}}\Pr(Y=k|X)\right] \leq \frac{K-1}{K}\end{aligned}$$

Solution: Under the 0-1 loss,

$$R(h) = \Pr(Y \neq h(X)) = \int_{\mathcal{X}} \Pr(Y \neq h(X)|X = x) f_X(x) dx.$$

Similarly, for any x, we want to find $h(x)$ that minimises $\Pr(Y \neq h(X)|X = x)$, or equivalently maximises $\Pr(Y = h(X) \mid X = x)$. This is given by

$$h^*(x) = \arg\max_{k \in \mathcal{Y}} \Pr(Y = k|X = x)$$

with associated risk

$$R(h^*) = 1 - \mathbb{E}\left[\max_{k \in \mathcal{Y}} \Pr(Y = k|X)\right]$$

Note that for any x, $\max_{k \in \mathcal{Y}} \Pr(Y = k|X = x) \geq 1/K$, giving the upper bound.

(d) Consider a binary classification problem ($\mathcal{Y} = \{1, -1\}$) with the 0-1 loss. Show that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

$$R(h^*) = \frac{1}{2} - \mathbb{E}\left[|\eta(X) - 1/2|\right]$$

$$R(h) - R(h^*) = \mathbb{E}\left[|2\eta(X) - 1| \mathbb{1}_{h^*(X) \neq h(X)}\right]$$

where $\eta(x) = \Pr(Y = 1 \mid X = x)$.

Solution: From the earlier part,

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

Additionally, using the identity

$$\max(a, b) = \frac{1}{2}(a + b) + \frac{1}{2}|a - b|$$

we obtain

$$\begin{aligned} R(h^*) &= 1 - \mathbb{E}\left[\frac{1}{2}(\eta(X) + 1 - \eta(X)) + \frac{1}{2}|2\eta(X) - 1|\right] \\ &= \frac{1}{2} - \mathbb{E}[|\eta(X) - 1/2|] \end{aligned}$$

Let h be some classifier. For any x,

$$\begin{aligned} \Pr(Y \neq h(X) \mid X = x) &= 1 - \Pr(Y = h(X) \mid X = x) \\ &= 1 - \Pr(Y = h(X) = 1 \mid X = x) - \Pr(Y = h(X) = -1 \mid X = x) \\ &= 1 - \mathbb{1}_{h(x)=1} \Pr(Y = 1|X = x) - \mathbb{1}_{h(x)=-1} \Pr(Y = -1|X = x) \\ &= 1 - \eta(x)\mathbb{1}_{h(x)=1} - (1 - \eta(x))(1 - \mathbb{1}_{h(x)=1}) \\ &= \eta(x) + (1 - 2\eta(x))\mathbb{1}_{h(x)=1} \end{aligned}$$

It follows that

$$\begin{aligned}\Pr(Y \neq h(X) \mid X = x) - \Pr(Y \neq h^*(X) \mid X = x) &= (2\eta(x) - 1)(\mathbb{1}_{h^*(x)=1} - \mathbb{1}_{h(x)=1}) \\ &= |2\eta(x) - 1|\mathbb{1}_{h(x)\neq h^*(x)}\end{aligned}$$

Taking the expectation with respect to X gives the result.

François Caron: caron@stats.ox.ac.uk

2. Consider a binary classification, where the joint distribution of $(X, Y) \in \mathbb{R} \times \{-1, 1\}$ is defined as follows. Y is uniformly distributed on $\{-1, 1\}$, that is $\Pr(Y = 1) = \Pr(Y = -1) = 1/2$, and

$$X|Y = y \sim \begin{cases} \mathcal{N}(\mu, \sigma^2) & \text{if } y = 1 \\ \mathcal{N}(-\mu, \sigma^2) & \text{if } y = -1 \end{cases}$$

for some parameters $\mu \geq 0$, $\sigma > 0$. Denote

$$g_y(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-y\mu)^2}{2\sigma^2}}$$

the conditional probability density function of X given $Y = y$.

(a) Let h^* be the Bayes classifier under a 0-1 loss. Show that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

where

$$\eta(x) = \left(1 + e^{-2x\mu/\sigma^2}\right)^{-1}.$$

Solution: From the previous question,

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

where

$$\eta(x) = \Pr(Y = 1 \mid X = x).$$

Applying Bayes' theorem gives

$$\eta(x) = \Pr(Y = 1 \mid X = x) = \frac{g_1(x)\Pr(Y = 1)}{g_1(x)\Pr(Y = 1) + g_{-1}(x)\Pr(Y = -1)} = \left(1 + e^{-2x\mu/\sigma^2}\right)^{-1}.$$

(b) Deduce that

$$h^*(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Solution: For any $\mu \geq 0$, $\sigma > 0$, η is a continuous, monotone increasing function, with $\eta(0) = 1/2$. The optimal classifier under the $0 - 1$ loss is therefore

$$h^*(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

(c) Show that the Bayes risk is

$$R(h^*) = 1 - \Phi\left(\frac{\mu}{\sigma}\right)$$

where Φ denotes the cumulative distribution function of the standard normal. How does the Bayes risk relate to the between-class and within-class variances of (X, Y)?

Solution: The associated Bayes risk is

$$\begin{aligned} R(h^*) &= \Pr(Y \neq h^*(X)) = \Pr(Y = 1, X < 0) + \Pr(Y = -1, X \geq 0) \\ &= \frac{1}{2}\left(\int_{-\infty}^{0} g_1(x)dx + \int_{0}^{\infty} g_{-1}(x)dx\right) \\ &= 1 - \Phi\left(\frac{\mu}{\sigma}\right) \end{aligned}$$

where Φ denotes the cumulative distribution function of the standard normal. The Bayes risk is a decreasing function of μ. When $\mu = 0$, the Bayes risk $R(h^*) = 1/2$ is maximal: the two classes overlap completely, and the optimal Bayes rule does not do better than random guess. The between-class variance is $\text{var}(\mathbb{E}[X|Y]) = \mu^2$ and the within-class variance is $\mathbb{E}[\text{var}(X|Y)] = \sigma^2$ (shown in the lectures). The ratio μ^2/σ^2 is therefore a measure of the separability of the classes. As the value μ/σ increases, the classes become more separated, and the Bayes risk decreases. The Bayes classifier achieves the lowest risk amongst all classifiers.

(d) Consider the classifier

$$h(x) = \begin{cases} 1 & \text{if } x \geq a \\ -1 & \text{if } x < a \end{cases}$$

for some $a \neq 0$. Give the expression of its risk $R(h)$ as a function of μ, σ, a and Φ. Check that $R(h) \geq R(h^*)$.

Solution:

Its risk is given by

$$R(h) = \frac{1}{2}\left(\int_{-\infty}^{a} g_1(x)dx + \int_{a}^{\infty} g_{-1}(x)dx\right) = 1 - \frac{1}{2}\left(\Phi\left(\frac{\mu - a}{\sigma}\right) + \Phi\left(\frac{\mu + a}{\sigma}\right)\right).$$

Differentiating with respect to a, we see that the minimum is obtained for $a = 0$.

3. Let $d = ((x_1, y_1), \ldots, (x_n, y_n))$ be a realisation from $D = ((X_1, Y_1), \ldots, (X_n, Y_n))$ where (X_i, Y_i) are iid with the same distribution as (X, Y). Let L be a loss function, and h be a fixed prediction rule, that does not depend on d. Assume $R(h) = \mathbb{E}[L(Y, h(X))] < \infty$. Let

$$\widehat{R}_n^{(d)}(h) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$$

be the empirical risk of h, which is a realisation of the random variable (as dependent on the random sample D)

$$\widehat{R}_n^{(D)}(h) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, h(X_i)).$$

Note: You may use any standard limit theorem or inequality for random variables. See Chapter 2 of Part A Probability for a refresher.

(a) Show that $\widehat{R}_n^{(D)}(h)$ is an unbiased estimator of $R(h)$.

(b) Show that $\widehat{R}_n^{(D)}(h)$ is a consistent estimator of $R(h)$.

Solution: Let $Z_i = L(Y_i, h(X_i))$. $Z_1, \ldots, Z_n$ are iid random variables with mean $R(h)$.

$$\mathbb{E}[\widehat{R}_n^{(D)}(h)] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[Z_i\right] = R(h)$$

Using the strong law of large numbers,

$$\widehat{R}_n^{(D)}(h) \to R(h)$$

almost surely as n tends to infinity, or $\Pr(\lim_{n\to\infty} \widehat{R}_n^{(D)}(h) = R(h)) = 1$. Hence $\widehat{R}_n^{(D)}(h)$ is a consistent estimator of $R(h)$. [Fine to use the weak law of large numbers instead].

(c) Assume $\sigma(h)^2 := \mathbb{E}[(L(Y, h(X)) - R(h))^2] < \infty$.

(i) Show that

$$\sqrt{n}(\widehat{R}_n^{(D)}(h) - R(h)) \to \mathcal{N}(0, \sigma(h)^2)$$

as n tends to infinity.

(ii) Show that, for any $\epsilon > 0$

$$\Pr(|\widehat{R}_n^{(D)}(h) - R(h)| > \epsilon) \leq \frac{\sigma(h)^2}{n\epsilon^2}.$$

(iii) If $\mathcal{Y} = \{1, \ldots, K\}$ and L is the $0-1$ loss, show that $\sigma(h) \leq 1$. Deduce a sufficient number n of examples so that

$$\Pr(|\widehat{R}_n^{(D)}(h) - R(h)| > \epsilon) \leq \delta,$$

as a function of $\epsilon > 0$ and $\delta > 0$.

François Caron: caron@stats.ox.ac.uk

Solution: Note that $\sigma(h)^2 = \text{var}(Z_i)$. $Z_1, \ldots, Z_n$ are iid with mean $R(h)$ and variance $\sigma(h)^2$. Using the Central Limit Theorem,

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} Z_i - R(h)\right) \to \mathcal{N}(0, \sigma(h)^2)$$

as n tends to infinity. For large n, the empirical risk is therefore normally distributed around the true risk, with an error term of order $\frac{\sigma(h)}{\sqrt{n}}$. Note that,

$$\begin{aligned}
\text{var}(\widehat{R}_n^{(D)}(h)) &= \text{var}\left(\frac{1}{n}\sum_{i=1}^{n} Z_i\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} \text{var}(Z_i) \quad \text{[by independence of the } Z_i\text{]} \\
&= \frac{\sigma(h)^2}{n}
\end{aligned}$$

Using Chebyshev inequality, we obtain the bound, or concentration inequality

$$\Pr(|\widehat{R}_n^{(D)}(h) - R(h)| > \epsilon) \leq \frac{\sigma(h)^2}{n\epsilon^2}.$$

Under the 0-1 loss,

$$\sigma(h)^2 = (1-R(h))^2 \Pr(Y \neq h(X)) + R(h)^2(1-\Pr(Y \neq X)) \leq \max(R(h)^2, (1-R(h))^2) \leq 1.$$

Hence we need $\delta \geq \frac{1}{n\epsilon^2}$ so $n \geq \frac{1}{\delta\epsilon^2}$. Much better bounds exist in the classification case with a bounded loss, such as Hoeffding's inequality. These bounds are covered in the Part C/MSc course on the Algorithmic Foundations of Learning.

4. Consider two univariate normal distributions $\mathcal{N}(\mu, \sigma^2)$ with known parameters $\mu_A = 10$ and $\sigma_A = 5$ for class A and $\mu_B = 20$ and $\sigma_B = 5$ for class B. Suppose class A represents the random score X of a medical test of normal patients and class B represents the score of patients with a certain disease. A patient is 100 times more likely to be healthy than to carry the disease.

   (a) Find the optimal decision rule h^* under the 0-1 loss for allocating a new observation x to either class A or B. Under h^*, what is the probability of misclassifying a healthy patient (A) as having the disease (B)? What is the probability of not detecting the disease (that is misclassifying a type B patient as A)? Given that a patient has the disease, what is the probability of not detecting it?

   Solution: The optimal decision for $X = x$ is to allocate to class

   $$\text{argmax}_{k \in \{A,B\}} \pi_k g_k(x).$$

François Caron: caron@stats.ox.ac.uk

The patients should be classified as healthy iff

$$\pi_A \frac{1}{\sqrt{2\pi}\sigma_A} \exp\left(-\frac{(x-\mu_A)^2}{2\sigma_A^2}\right) \geq \pi_B \frac{1}{\sqrt{2\pi}\sigma_B} \exp\left(-\frac{(x-\mu_B)^2}{2\sigma_B^2}\right),$$

that is, using $\sigma_A = \sigma_B$, iff

$$-2\sigma_A^2 \log(\pi_A/\pi_B) + (x-\mu_A)^2 \leq (x-\mu_B)^2.$$

The decision boundary is attained for equality, that is if x fulfills

$$2x(\mu_B - \mu_A) + \mu_A^2 - \mu_B^2 - 2\sigma_A^2 \log(\pi_A/\pi_B) = 0.$$

For the given values, this implies that the decision boundary is at

$$x = (50 \log 100 - 100 + 400)/(2 \cdot 10) \approx 26.51,$$

that is all patients with a test score above 26.51 are classified as having the disease. $h^*(x) = B$ if $x > 26.51$ and $h^*(x) = A$ otherwise. We have

$$\begin{aligned}
\Pr(Y = A, h^*(X) = B) &= \Pr(Y = A, X > 26.51) = \Pr(X > 26.51 | Y = A)\pi_A \\
&= \pi_A(1 - \Phi(\frac{26.51 - \mu_A}{\sigma_A})) \simeq 0.0106 \\
\Pr(Y = B, h^*(X) = A) &= \Pr(Y = B, X < 26.51) = \Pr(X < 26.51 | Y = B)\pi_B \\
&= \pi_B \Phi(\frac{26.51 - \mu_B}{\sigma_B}) \simeq 0.0089 \\
\Pr(h^*(X) = A \mid Y = B) &= \Phi(\frac{26.51 - \mu_B}{\sigma_B}) \simeq 0.9035
\end{aligned}$$

(b) Repeat (a) if the cost of a false negative (allocating a sick patient to group A) is $\theta > 1$ times that of a false positive (allocating a healthy person to group B). Describe how the rule changes as θ increases. For which value of θ are 84.1% of all patients with disease correctly classified?

Solution: For any $x \in \mathbb{R}$, the optimal decision $h^*(x)$ minimises $\mathbb{E}\left[L(Y, h(X))|X = x\right]$ where now the loss is defined as

$$L(y, h(x)) = \begin{cases} c\theta & \text{if } h(x) = A \text{ and } y = B \\ c & \text{if } h(x) = B \text{ and } y = A \\ 0 & \text{if } y = h(x) \end{cases}$$

for some constants $c > 0$ and $\theta > 1$. The conditional risk is therefore

$$\mathbb{E}\left[L(Y, h(X))|X = x\right] = c\theta \Pr(Y = B|X = x)\mathbb{1}_{h(x)=A} + c \Pr(Y = A|X = x)\mathbb{1}_{h(x)=B}$$

It is hence optimal to choose class A (healthy) over class B if and only if

$$\Pr(Y = A|X = x) \geq \theta \Pr(Y = B|X = x).$$

Using the same argument as above, the patients should be classified as healthy now iff (ignoring again the common denominator $\sum_{k\in\{A,B\}} \pi_k g_k(x)$),

$$\pi_A \frac{1}{\sqrt{2\pi}\sigma_A} \exp(-\frac{(x-\mu_A)^2}{2\sigma_A^2}) \geq \theta\pi_B \frac{1}{\sqrt{2\pi}\sigma_B} \exp(-\frac{(x-\mu_B)^2}{2\sigma_B^2}).$$

The decision boundary is now attained if x fulfills

$$2x(\mu_B - \mu_A) + \mu_A^2 - \mu_B^2 - 2\sigma_A^2 \log(\pi_A/(\theta\pi_B)) = 0.$$

The decision boundary

$$t_\theta = \frac{\mu_B^2 - \mu_A^2 + 2\sigma_A^2 \log(\pi_A/(\theta\pi_B))}{2(\mu_B - \mu_A)} \tag{1}$$

and $h^*(x) = B$ if $x > t_\theta$ and A otherwise. For increasing values of θ, the threshold t_θ decreases and patients with smaller test scores are classified as having the disease (e.g. further screened).

We want

$$\Pr(h^*(X) = B \mid Y = B) = \Pr(X > t_\theta \mid Y = B) = 1 - \Phi((t_\theta - \mu_B)/\sigma_B) = 0.841.$$

This gives

$$t_\theta = \mu_B + \sigma_B \Phi^{-1}(0.159) \approx 15$$

which, combined with (1), gives

$$\theta = 100.$$

5. Suppose we have a binary classification problem, i.e., $\mathcal{Y} = \{-1, 1\}$, and a 2-dimensional input variable X. Assume that the sample means of the two groups are at $\hat{\mu}_{-1} = (-1, -1)^\top$ and $\hat{\mu}_1 = (1, 1)^\top$ respectively. The estimated prior class probabilities $\hat{\pi}_1$ and $\hat{\pi}_{-1}$ are equal.

   (a) Applying LDA, the covariance matrix is estimated to be, for some value of $0 \leq \rho < 1$,

   $$\hat{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

   Find the decision boundary as a function of ρ.

Solution: In LDA, decision boundary is $a_\star + b_\star^\top x$, where

$$a_\star = a_1 - a_{-1} = \log(\hat{\pi}_1) - \frac{1}{2}\hat{\mu}_1^T\hat{\Sigma}^{-1}\hat{\mu}_1 - \log(\hat{\pi}_{-1}) + \frac{1}{2}\hat{\mu}_{-1}^T\hat{\Sigma}^{-1}\hat{\mu}_{-1}.$$

Hence $a_\star = 0$. Further, $b_\star = b_1 - b_{-1}$ is

$$b_\star = \hat{\Sigma}^{-1}(\mu_1 - \mu_{-1}) = \hat{\Sigma}^{-1}(2,2)^\top = 2/(1+\rho)(1,1)^\top,$$

Class 1 is chosen over class -1 for $x = (x_1, x_2)^\top$ if and only if $a_\star + b_\star^\top x > 0$, that is iff

$$\frac{2}{1+\rho}(x_1 + x_2) > 0.$$

Equivalently, iff

$$x_1 + x_2 > 0,$$

which could have been guessed as the solution initially.

(b) Suppose instead that, we model each class with its own covariance matrix. We estimate the covariance matrices for group -1 as

$$\hat{\Sigma}_{-1} = \begin{pmatrix} 5 & 0 \\ 0 & 1/5 \end{pmatrix},$$

and for group 1 as

$$\hat{\Sigma}_1 = \begin{pmatrix} 1/5 & 0 \\ 0 & 5 \end{pmatrix}.$$

Describe the decision rule and draw a sketch of it in the two-dimensional plane.

Solution: In this case, discriminant function is quadratic, i.e., for $k \in \{-1, 1\}$,

$$\log \hat{\pi}_k \hat{g}_k(x) = \text{const} + a_k + b_k^\top x + x^\top c_k x,$$

where

$$\begin{aligned} a_k &= \log \hat{\pi}_k - \frac{1}{2}\log\left|\hat{\Sigma}_k\right| - \frac{1}{2}\hat{\mu}_k^\top\hat{\Sigma}_k^{-1}\hat{\mu}_k, \\ b_k &= \hat{\Sigma}_k^{-1}\hat{\mu}_k, \\ c_k &= -\frac{1}{2}\hat{\Sigma}_k^{-1}. \end{aligned}$$

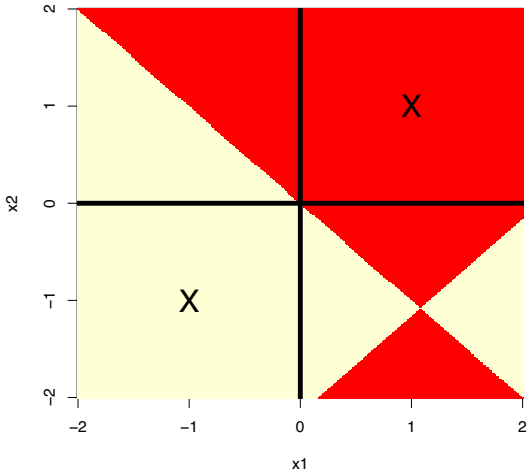
Substituting gives

$$\begin{aligned} a_\star &= a_1 - a_{-1} = 0, \\ b_\star &= b_1 - b_{-1} = (5, 1/5)^\top - (-1/5, -5)^\top = \frac{26}{5}(1,1)^\top, \\ c_\star &= c_1 - c_{-1} = \begin{pmatrix} -12/5 & 0 \\ 0 & 12/5 \end{pmatrix}. \end{aligned}$$

Now, class 1 is chosen if and only if $b_\star^\top x + x^\top c_\star x > 0$, or equivalently

$$\begin{aligned} & \frac{26}{5}(x_1 + x_2) - \frac{12}{5}\left((x_1)^2 - (x_2)^2\right) > 0 \\ \Leftrightarrow \quad & (x_1 + x_2)\left(\frac{13}{6} - x_1 + x_2\right) > 0. \end{aligned}$$

Thus, decision boundaries are formed by hyperplanes, even though we are using QDA (which typically produces quadratic decision boundaries). The decision boundaries are shown in the figure below, along with the group means.



6. The Wine dataset contains the quantities of $p = 13$ constituents found in $n = 178$ Italian wines. The wines come from the same region, but are derived from $K = 3$ different cultivars. A description of the dataset is given at `https://archive.ics.uci.edu/ml/datasets/Wine`. The goal is to build a classifier for predicting the cultivar of a wine based on its constituents. Here is the code to load the data in Python.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler # to normalise the data
from sklearn.datasets import load_wine
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Load wine data
(X,y) = load_wine(return_X_y = True, as_frame = True)

(n, p) = X.shape # number of examples and dimension of each example
K = y.unique().size # number of classes
print('n=', n, ', p=', p, ', K=', K)
Xy = pd.concat([X, y], axis=1) # X and y in a single data frame
```

   (a) Apply LDA to this dataset. Report the training error for the learned classifier.

   (b) Visualise the coefficients b_k of the decision boundaries.

   (c) Plot the projections of the data in the 2D LDA discriminant space.

Solution: See Python notebook.

François Caron: caron@stats.ox.ac.uk

7. (optional) We consider a dataset of $n = 1797$ 8-by-8 images (hence $p = 64$) of handwritten digits ($K = 10$ classes, labelled 0 to 9). The following Python code loads the dataset, plots some sample images, and splits the dataset into a training set and a test set. The objective is to train a classifier on the training set, and predict the labels of images on the test set.

```
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
# Load data
(X, y) = load_digits(return_X_y=True, as_frame=True, n_class = 10)
# Plot some images in the dataset
for i in range(30):
    plt.subplot(3, 10, i + 1)
    image = X.loc[i,:].to_numpy()
    plt.imshow(image.reshape(8, 8), cmap=plt.cm.gray)
    plt.title(y[i])
    plt.xticks(())
    plt.yticks(())
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

   (a) Train the LDA classifier on the training set. Report the accuracy (proportion of correctly classified images) of the classifier on both the training set and the test set.

   (b) Show the projection of the training data in the 2D LDA discriminant space.

   (c) For comparison, show the projection of the training data onto the first two principal components of PCA

   (d) Train the QDA classifier on the training set, with no regularisation. Report the accuracy of the classifier on the training and test set.

   (e) Retrain the QDA classifier, now using some regularisation. Report the accuracy of the classifier on the training and test set.

   (f) Train the Naive Bayes classifier with Gaussian likelihoods on the training set. Report the accuracy of the classifier on the training and test set.

Solution: See Python notebook.

François Caron: caron@stats.ox.ac.uk

Statistical Machine Learning

Sheet 3 — HT21

Overfitting, Regularisation, ROC curves, Optimisation, Linear Classifiers

1. The binary Naive Bayes classifier has interesting connections to the logistic regression classifier. You will show that, under certain assumptions, the conditional distribution $\Pr(Y = 1 \mid X = x)$ for Naive Bayes has a similar form to logistic regression. You will then derive the maximum likelihood parameter estimates under these assumptions.

   Suppose $X = (X_1, \ldots, X_p)^\intercal$ is a continuous random vector in $\mathbb{R}^p$, and Y is a binary random variable with values in $\{-1, 1\}$ representing the class labels. Let $\pi_1 = \Pr(Y = 1)$ and g_k denote the conditional probability density function of X given that $Y = k$. We make the Naive Bayes assumption that

$$g_k(x) = \prod_{j=1}^{p} g_{kj}(x_j; \mu_{kj}, \sigma_j^2)$$

   where $x = (x_1, \ldots, x_p)^\intercal \in \mathbb{R}^p$ and

$$g_{kj}(x_j; \mu_{kj}, \sigma_j^2) = (2\pi\sigma_j^2)^{-1/2} e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma_j^2}}.$$

   That is, conditionally on $Y = k$, $X_1, \ldots, X_p$ are independent, and X_j given $Y = k$ follows a normal distribution with mean μ_{kj} and variance σ_j^2. Note that the variance parameter depends on the input dimension j but not on the class k.

   (a) For a given $x = (x_1, \ldots, x_p)^\intercal \in \mathbb{R}^p$, show that

$$\Pr(Y = 1 | X = x) = \frac{1}{1 + \exp(-\beta_0 - \sum_{j=1}^{p} \beta_j x_j)}.$$

   where you should give the explicit form of $\beta_0, \ldots, \beta_p$ in terms of π_1, μ_{kj}, σ_j, for $j = 1, \ldots, p$ and $k \in \{-1, 1\}$.

   (b) This part is independent from part (a).

   Suppose a training set with n examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ is given, where $x_i = (x_{i1}, \ldots, x_{ip})^\top$ is a p-dimensional input vector, and $y_i \in \{-1, 1\}$ is its corresponding label. Using the given naive Bayes assumption, provide the maximum likelihood estimate for the parameters $\theta = (\pi, (\mu_{1,j}, \mu_{-1,j}, \sigma_j^2)_{j=1,\ldots,p})$.

François Caron: caron@stats.ox.ac.uk

2. Assume we trained a classifier using a dataset $d = ((x_1, y_1), \ldots, (x_n, y_n))$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, 2, ..., K\}$. We are interested in classifying a new input vector x_{n+1}. However, we have only been able to collect $p - 1$ features, say $(x_{n+1,2}, ..., x_{n+1,p})$ and $x_{n+1,1}$ is missing. Explain whether or not it is possible to use the trained classifier to classify this incomplete input vector in the cases listed below. If it is possible, how do you classify the incomplete test vector?

   Note: You do not need to calculate any integrals in this question. For generative classifiers, $g_k(x)$ denotes the conditional class probability mass/density function of X given $Y = k$.

   (a) A naïve Bayes model, with

   $$g_k(x_i) = \prod_{j=1}^{p} g_{kj}(x_{ij}; \theta_{kj}),$$

   i.e. conditioned upon $Y_i = k$, you assume that the different dimensions of the input are independent and X_{ij} given $Y_i = k$ has probability mass/density function $g_{kj}(x_{ij}; \theta_{kj})$.

   (b) An LDA model, i.e.

   $$g_k(x_i) = \varphi\left(x_i; \mu_k, \Sigma\right)$$

   where $\varphi\left(x; \mu, \Sigma\right)$ denotes the pdf of the multivariate normal with mean μ and covariance matrix Σ, evaluated at x.

   (c) Generally, what condition on the conditional pdf/pmf $g_k(x)$ would allow easy classification (i.e, without numerical integration) in the presence of missing features for generative classifiers like LDA or naïve Bayes?

   (d) A binary logistic regression model

   $$\Pr(Y = 1 | X = x) = \text{sig}(a + b^\intercal x).$$

François Caron: caron@stats.ox.ac.uk

3. Let $d_{\text{train}} = ((x_1, y_1), \ldots, (x_n, y_n))$ and $d_{\text{test}} = ((\widetilde{x}_1, \widetilde{y}_1), \ldots, (\widetilde{x}_m, \widetilde{y}_m))$ be respectively the training dataset and test dataset. These are realisations of the random samples D_{train} and D_{test}, which are composed of iid random variables with the same distribution as (X, Y). Let L be a loss function, and denote $R(h) = \mathbb{E}[L(Y, h(X))]$ be the associated population risk. Denote

$$\widehat{R}_n^{(d_{\text{train}})}(h) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))$$

be the empirical risk of a prediction rule h on the training set. Similarly, let

$$\widehat{R}_m^{(d_{\text{test}})}(h) = \frac{1}{m} \sum_{i=1}^{m} L(\widetilde{y}_i, h(\widetilde{x}_i))$$

be the empirical risk of a prediction rule h on the test set. Let $\mathcal{H}$ be some class of prediction rules. Let

$$\widehat{h}^{(d_{\text{train}})} = \arg\min_{h \in \mathcal{H}} \widehat{R}_n^{(d_{\text{train}})}(h)$$

be the empirical risk minimiser and

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} R(h)$$

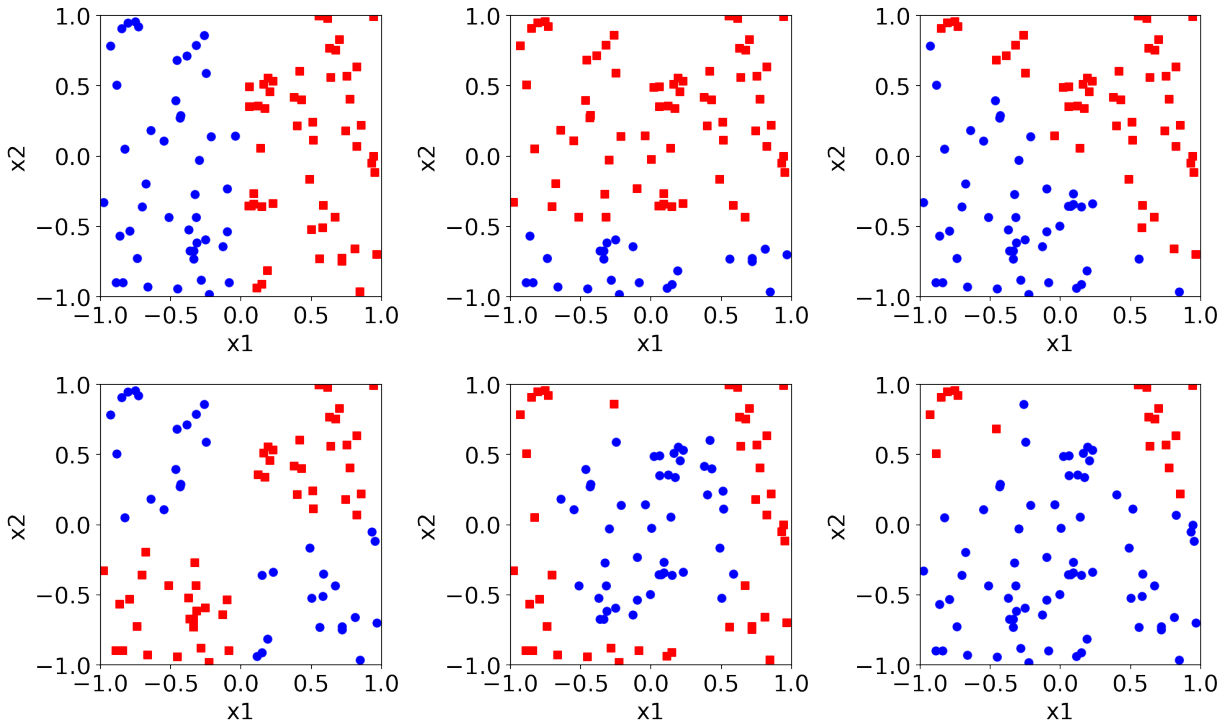
the best prediction rule in the class $\mathcal{H}$.

Show that

$$\mathbb{E}[\widehat{R}_n^{(D_{\text{train}})}(\widehat{h}^{(D_{\text{train}})})] \leq R(h^*_{\mathcal{H}}) \leq \mathbb{E}[\widehat{R}_m^{(D_{\text{test}})}(\widehat{h}^{(D_{\text{train}})})]$$

where the expectations are taken with respect to the training sample D_{train} and test sample D_{test}.

4. For each of the datasets below, find a potentially non-linear function $\phi : \mathbb{R}^2 \to \mathbb{R}^M$ which makes the transformed inputs linearly separable (limit yourself to polynomial expansions with interaction terms). That is, there exists a parameter β such that the discriminant function $f(x) = \phi(x)^\intercal \beta$ perfectly separates the data.



5. (a) Show that if g_1 and g_2 are convex functions and c_1, c_2 are nonnegative scalars, $c_1 g_1 + c_2 g_2$ is convex.

   (b) Show that if $g : \mathbb{R} \to \mathbb{R}$ is convex, $\psi(\beta) = g(a^\intercal \beta + b)$ is convex where a, β are p-dimensional vectors and b is a scalar.

   (c) Show that the following functions are convex: $g_1(x) = |x|$, $g_2(x) = e^{-x}$, $g_3(x) = (1-x)^2$, $g_4(x) = \log(1 + e^{-x})$, $g_5(x) = \max(0, -x)$.

   (d) What can you conclude about the objective functions of the least-squares, perceptron and logistic regression (linear) classifiers, with a L2 regularisation?

6. Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \mathbb{R}$ the target space. Consider the class of linear prediction rules $h(x) = \phi(x)^\intercal \beta$ with input expansion $\phi : \mathcal{X} \to \mathbb{R}^p$ and $\beta \in \mathbb{R}^p$. The regularised empirical risk minimiser, with L2 penalty, is $\widehat{h}(x) = \phi(x)^\intercal \widehat{\beta}$ where $\widehat{\beta}$ minimises the ridge regression objective function

$$\begin{aligned} J(\beta) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \phi(x_i)^\intercal\beta)^2 + \frac{\lambda}{n}\|\beta\|^2 \\ &= \frac{1}{n}\|\,\mathbf{y} - \mathbf{\Phi}\beta\|^2 + \frac{\lambda}{n}\|\beta\|^2 \end{aligned} \tag{1}$$

where $\lambda > 0$ is the regularisation parameter.

(a) Show that $\widehat{\beta}$ can be interpreted as the maximum a posteriori (that is, the maximum of the posterior probability density function) of the parameter β under the model

$$\mathbf{y} \mid \beta \sim \mathcal{N}(\mathbf{\Phi}\beta, \sigma^2 I_n), \qquad \beta \sim \mathcal{N}(0, \tau I_p)$$

where $\tau > 0$, $\sigma > 0$, and I_k denotes the k-by-k identity matrix. Find the relationship between the regularisation parameter λ and the variances τ and σ^2.

(b) Show that $\widehat{\beta}$ can also be obtained by applying ordinary least squares regression (that is, without regularisation) on an augmented dataset $\widetilde{\mathbf{y}}$, $\widetilde{\mathbf{\Phi}}$, that depends on $\mathbf{y}$, $\mathbf{\Phi}$ and λ.

(c) Give the gradient descent algorithm to minimise (1).

François Caron: caron@stats.ox.ac.uk

7. We consider a dataset originally collected by the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset consists of several medical predictors (Blood pressure, body mass index, age, etc) and a binary output, indicating if the person has diabete or not. All patients here are female, at least 21 years old and of Pima Indian heritage. The objective is to derive a classifier for predicting if a given person is diabetic or not, based on its medical inputs. The dataset can be downloaded from `https://www.kaggle.com/kumargh/pimaindiansdiabetescsv`.

   The following code loads the dataset and removes missing data. The cleaned dataset has $n = 724$ patients with $p = 6$ inputs. The data is split between a training and a test set.

```
# Standard libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
# Train-test and preprocessing functions
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Classifiers
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
# Metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

# Download the data from
# https://www.kaggle.com/uciml/pima-indians-diabetes-database
filename = '../datasets/pimaIndians/diabetes.csv'
diabetes = pd.read_csv(filename)

# Missing values indicated by 0 - replace by nan
for name in diabetes.keys()[1:-1]:
# for features other than pregnancies, 0 indicate a missing value, so replace by nan
    diabetes[name] = diabetes[name].replace({0: np.nan})
diabetes.isna().sum()

 # remove the dimensions with large number of missing data
df = diabetes.drop(columns=['SkinThickness', 'Insulin'])
df = df.dropna() # remove examples with missing data
# X and y data
X = df[df.keys()[0:-1]]
y = df[df.keys()[-1]]
```

```
(n,p) = X.shape
print('n =', n, ', p =', p)

# Split training/test set
X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=0.25, random_state=12)
```

(a) Visualise the data using a pairplot

(b) Standardise the training data, and apply the same transformation to the test data (use the function `StandardScaler`)

(c) For each of the following plug-in classifiers: Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naive Bayes, Logistic Regression:

- Estimate the parameters of the classifiers on the training data
- Under the 0-1 loss, report the accuracy and confusion matrix on the training and test datasets
- Plot the ROC curve and calculate the Area Under the Curve (AUC) on both the training and test set

(d) (Optional): Try to apply some input transformation, adding interaction terms. Does it improve the performances in terms of AUC?

SB 2.2 Statistical Machine Learning Sheet 3

Ruizhan Ma

1. (a) By Bayes rule:

$$\mathbb{P}(Y=1 \mid X=x) = \frac{\mathbb{P}(Y=1)\,\mathbb{P}(X=x \mid Y=1)}{\mathbb{P}(X=x)}$$

$$= \frac{\mathbb{P}(Y=1)\,\mathbb{P}(X=x \mid Y=1)}{\mathbb{P}(X=x \mid Y=0)\,\mathbb{P}(Y=0) + \mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)}$$

$$= \frac{1}{1 + \frac{\mathbb{P}(X=x \mid Y=0)\,\mathbb{P}(Y=0)}{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)}}$$

since $$\frac{\mathbb{P}(X=x \mid Y=0)\,\mathbb{P}(Y=0)}{\mathbb{P}(X=x \mid Y=1)\,\mathbb{P}(Y=1)} = \exp\left[\left(\ln\frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)}\right) + \ln\left(\frac{\mathbb{P}(X=x \mid Y=0)}{\mathbb{P}(X=x \mid Y=1)}\right)\right]$$

$$= \exp\left[\ln\frac{1-\pi_1}{\pi_1} + \sum_j \ln\left(\frac{\mathbb{P}(x_j \mid Y=0)}{\mathbb{P}(x_j \mid Y=1)}\right)\right]$$

where $$\sum_j \ln\left(\frac{\mathbb{P}(x_j \mid Y=0)}{\mathbb{P}(x_j \mid Y=1)}\right) = \sum_j \ln\frac{\exp(-(x_j-\mu_{0j})^2/2\sigma_j^2)}{\exp(-(x_j-\mu_{1j})^2/2\sigma_j^2)}$$

$$= \sum_j \frac{(x_j-\mu_{1j})^2 - (x_j-\mu_{0j})^2}{2\sigma_j^2}$$

$$= \sum_j \left(\frac{\mu_{0j}-\mu_{1j}}{\sigma_j^2}x_j + \frac{\mu_{1j}^2-\mu_{0j}^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \mathbb{P}(Y=1 \mid X=x) = \frac{1}{1+\exp\left[\sum_j\left(\frac{\mu_{0j}-\mu_{1j}}{\sigma_j^2}x_j + \frac{\mu_{1j}^2-\mu_{0j}^2}{2\sigma_j^2}\right) + \ln\frac{1-\pi_1}{\pi_1}\right]}$$

Let $-\frac{\mu_{0j}-\mu_{1j}}{\sigma_j^2} = \beta_j$, $-\sum_j \frac{\mu_{1j}^2-\mu_{0j}^2}{2\sigma_j^2} - \ln\frac{1-\pi_1}{\pi_1} = \beta_0$

Then we have the form

$$\mathbb{P}(Y=1 \mid X=x) = \frac{1}{1+\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)} \quad \square$$

b) $l(\pi, (\mu_{1j}, \mu_{-1j}, \sigma_j^2)_{j=1\dots p}) = \sum_{i=1}^{n} \left(\log(\pi_{y_i}) + \log(g_{y_i}(x_i)) \right)$

$$= \sum_{k=1}^{K} n_k \log(\pi_k) + \sum_{k=1}^{K} \sum_{i:y_i=k} \log(g_k(x_i))$$

$$= \sum_{k=1}^{K} n_k \log(\pi_k) + \sum_{j=1}^{p} \sum_{k=1}^{K} \sum_{i:y_i=k} \log g_{kj}(x_{ij}, \theta_{kj})$$

$$\hat{\theta} = \operatorname{argmax} \sum_{i:y_i=k} \log g_{kj}(x_{ij}, \theta_{kj}) = \operatorname{argmax} \sum_{i:y_i=k} -\frac{1}{2}\log(2\pi\sigma_j^2) - \frac{(x_{ij}-\mu_{kj})^2}{2\sigma_j^2}$$

$\Rightarrow$ $\hat{\pi}_1 = n_1/n$ $\quad n_1$: # y_i s.t. $y_i = 1$

$$\hat{\mu}_{1j} = \frac{1}{n_1} \sum_{i:y_i=1} x_{ij} \quad ; \quad \hat{\mu}_{-1j} = \frac{1}{n_{-1}} \sum_{i:y_i=-1} x_{ij}$$

$$\hat{\sigma}_j = \frac{1}{n} \sum_i \sum_k (x_{ij} - \hat{\mu}_{kj})^2 \, \mathbb{1}\{y_i = k\}$$

2.

(a) $g_k(x_i) = \prod_{j=1}^{p} g_{kj}(x_{ij}, \theta_{kj})$

We can classify with incomplete test vector,

Conditional on $Y_i = k$, assume the different dimensions of input are iid.

Then we can ignore feature 1

$$\hat{P}(Y = k \mid X) \propto \pi_k \prod_{j=2}^{p} g_{kj}(x_{ij} \mid \theta_{kj}) \qquad \pi_k = \hat{P}(Y = k)$$

(b) $g_k(x_i) = \varphi(x_i; \mu_k, \Sigma)$

We can classify with incomplete test vector,

$$\hat{P}(Y = k \mid x_{n+1,2}, \dots x_{n+1,p}) \propto \pi_k \cdot g_k(x_{n+1,2} \dots x_{n+1,p})$$

where $g_k(x_{n+1,2} \dots x_{n+1,p})$ is a multivariate normal with mean $\mu = (\mu_{k2}, \dots \mu_{kn})$

$$\Sigma = \begin{pmatrix} & \\ & \end{pmatrix}_{(p-1)\times(p-1)}$$

Σ' is Σ dropping first column and first row,

(c) To be able to classify with incomplete data, we need to be able to compute

$$g_k(x_{obs}, \theta_k) = \int \hat{P}(x_{miss}, x_{obs}; \theta_k)\, dx_{miss}$$

$$= \int g_k(x)\, dx_{miss}$$

(d) Since $sig(a + b^T x) = \frac{1}{1 + \exp(-a + b^T x)}$

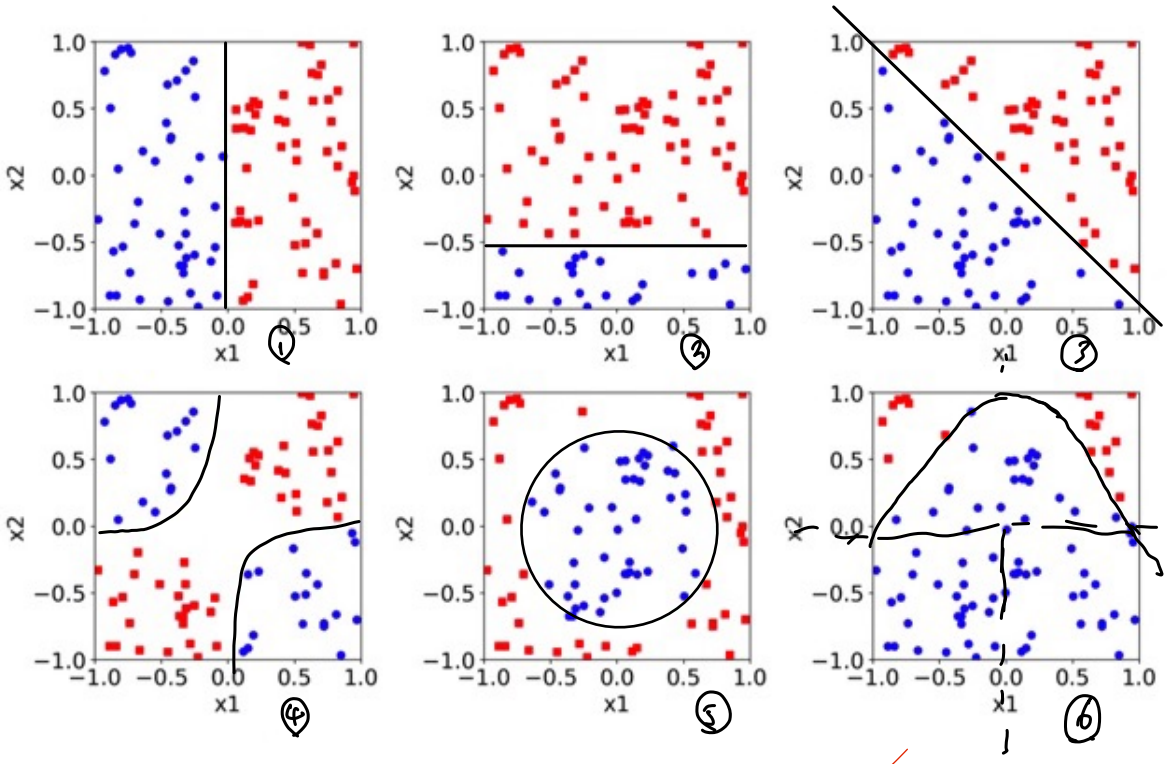
Sigmoid function is not a distribution, so when we have incomplete data, we can't classify.

3. To show: $\mathbb{E}[\hat{R}_n(\hat{h}^{(D_{train})})] \le R(h_H^*) \le \mathbb{E}[\hat{R}_m(\hat{h}^{(D_{train})})]$

Proof:

Since $\mathbb{E}[\hat{R}_n(\hat{h}^{(D_{train})})] = R_n(\hat{h}^{(D_{train})})$

4.



① , let $\phi(x) = 10\, x_1$

② : let $\phi(x) = 10\,(x_2 - 0.5)$

③ : let $\phi(x) = 10\,(x_2 + x_1)$

④ : let $\phi(x) = 10\,(x_1 x_2)$

⑤ : let $\phi(x) = x_1^2 + x_2^2 - \frac{3}{4}$

⑥ : let $\phi(x) = 10 x_2 + 20\,[x_2 - (x_1^2 - 1)]$

if decision is whether $f > 0$, ϕ also needs a constant term

5. (a) g_1, g_2 are convex, let $g_3 = c_1 g_1 + c_2 g_2$

Proof: $\forall u, v$, $\alpha \in [0,1]$

$$\begin{aligned} & g_3(\alpha u + (1-\alpha)v) \\ = \; & c_1 g_1(\alpha u + (1-\alpha)v) + c_2 g_2(\alpha u + (1-\alpha)v) \\ \le \; & c_1 \alpha g_1 u + c_1(1-\alpha) g_1(v) + c_2 \alpha g_2(u) + c_2(1-\alpha) g_2(v) \\ = \; & \alpha g_3(u) + (1-\alpha) g_3(v) \end{aligned}$$

□

(b) $g: \mathbb{R} \to \mathbb{R}$ is convex.

$$\psi(\beta) = g(a^T \beta + b)$$

Therefore we have:

$$\begin{aligned} & \psi(\alpha u + (1-\alpha)v) \\ = \; & g(a^T[\alpha u + (1-\alpha)v] + b) \\ = \; & g(a^T(\alpha u) + a^T(1-\alpha)v + b) \\ = \; & g(\alpha a^T u + (1-\alpha) a^T v + b) \\ \le \; & \alpha g(a^T u) + (1-\alpha) g(a^T v) \end{aligned}$$

□.

(c) $g_1(x) = |x|$ Since $g_1(\alpha u + (1-\alpha)v)$

$$= |\alpha u + (1-\alpha)v| \le |\alpha u| + |(1-\alpha)v| = \alpha g_1(u) + (1-\alpha) g_1(v)$$

□

$g_2(x) = e^{-x}$ Since $g_2''(x) = e^{-x} \ge 0$

Therefore $g_2(x) = e^{-x}$ convex □.

$g_3(x) = (1-x)^2$ Since $g_3''(x) = 2 \ge 0$

Therefore $g_3(x) = (1-x)^2$ convex □.

$g_4(x) = \log(1+e^{-x})$ Since $g_4'(x) = \dfrac{-e^{-x}}{1+e^{-x}}$

$$g_4''(x) = \frac{e^{-x}(1+e^{-x}) + e^{-x}(-e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} \ge 0$$

Therefore $g_4(x)$ convex. □.

$g_5(x) = \max(0, -x)$ $\quad g_5(\alpha u + (1-\alpha)v) = \max(0, -\alpha u - (1-\alpha)v)$

$\leq \max(0, -\alpha u) + \max(0, -(1-\alpha)v)$

$= \alpha g_5(u) + (1-\alpha) g_5(v)$ □

d).

6. $J(\beta) = \frac{1}{n}\|y - \Phi\beta\|^2 + \frac{\lambda}{n}\|\beta\|^2$

(a) We have $L(\beta|Y) \propto f(Y|\beta) \times \pi(\beta)$

$\propto (2\pi\sigma^2)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}(y-\Phi\beta)^T(y-\Phi\beta)\right]$

$\times (2\pi\tau)^{-p/2} \exp\left[-\frac{1}{2\tau}\beta^T\beta\right]$

$\Rightarrow l(\beta|Y) \propto -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y-\Phi\beta)^T(y-\Phi\beta)$

$- \frac{p}{2}(2\pi\tau) - \frac{1}{2\tau}\beta^T\beta$

Differentiate with respect to β.

$\frac{\partial l}{\partial \beta} = -\frac{1}{2\sigma^2}(2\Phi^T\Phi\beta - 2\Phi^T y) - \frac{1}{2\tau}\cdot 2\beta$

$= \frac{1}{\sigma^2}(\Phi^T y - \Phi^T\Phi\beta) - \frac{\beta}{\tau}$

$\frac{\partial l}{\partial \beta} = 0 \Rightarrow \frac{1}{\sigma^2}\Phi^T y = \left(\frac{1}{\tau} + \frac{\Phi^T\Phi}{\sigma^2}\right)\beta$

$\Rightarrow \hat{\beta} = \frac{\Phi^T y}{\frac{\sigma^2}{\tau} + \Phi^T\Phi}$ Don't write this! $\left(\frac{\sigma^2}{\tau} + \Phi^T\Phi\right)^{-1}\Phi^T y$.

(b) $J(\beta) = \frac{1}{n}\left(\beta^T\Phi^T\Phi\beta - 2(\Phi^T y)^T\beta + \lambda\beta^T\beta\right) + \text{const}$,

$\Rightarrow \frac{\partial J(\beta)}{\partial \beta} = \frac{1}{n}(2\Phi^T\Phi\beta - 2\Phi^T y + 2\lambda\beta)$

$\frac{\partial J(\beta)}{\partial \beta} = 0 \Rightarrow \hat{\beta} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$

$\hat{\beta} = \arg\max J(\beta)$

×

(c) Gradient descent algorithm to minimize

$$J(\beta) = \frac{1}{n}\|y - \Phi\beta\|^2 + \frac{\lambda}{n}\|\beta\|^2$$

$$\nabla_\beta J(\beta) = \frac{1}{n}\, 2\Phi^T(\Phi\beta - y) + 2\lambda\beta$$

$$\beta^{t+1} = \beta^t - \frac{1}{n}\left[2\Phi^T(\Phi\beta^t - y) + 2\lambda\beta^t\right]$$

① Initialize $\beta^{(0)} = 0$, set $t = 0$

② Repeat until convergence.

1. $\nabla_\beta J(\beta^t)$
2. update parameters:

$$\beta^{t+1} = \beta^t - \frac{1}{n}\left[2\Phi^T(\Phi\beta^t - y) + 2\lambda\beta^t\right]$$

3. set $t \leftarrow t+1$

Q7.

In [1]:
```
# Standard libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
# Train-test and preprocessing functions

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Classifiers
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression

# Metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

# For saving the figures
svfigs = False # True if you want to save figures
svdir = './figures/' # directory to save the figures
if svfigs: # set a higher resolution if saving the figures
    plt.rcParams['figure.dpi'] = 150
else:
    plt.rcParams['figure.dpi'] = 75

# Download the data from
# https://www.kaggle.com/uciml/pima-indians-diabetes-database
filename = '../datasets/pimaIndians/diabetes.csv'
diabetes = pd.read_csv('/Users/pan/Documents/Oxford/Yr3/HT/SB2.2 Statistica
```

In [2]:
```
diabetes
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
0	1	85	66	29	0	26.6	
1	8	183	64	0	0	23.3	
2	1	89	66	23	94	28.1	
3	0	137	40	35	168	43.1	
4	5	116	74	0	0	25.6	
...	...	...	...	...	...	...	
762	10	101	76	48	180	32.9	
763	2	122	70	27	0	36.8	
764	5	121	72	23	112	26.2	
765	1	126	60	0	0	30.1	
766	1	93	70	31	0	30.4	

767 rows × 9 columns

In [3]:
```
# Missing values indicated by 0 - replace by nan
for name in diabetes.keys()[1:-1]:
    diabetes[name] = diabetes[name].replace({0: np.nan})
diabetes.isna().sum()
# for features other than pregnancies, 0 indicate a missing value, so repl
```

Out[3]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     373
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Class                         0
dtype: int64
```

In [4]:
```
 # remove the dimensions with large number of missing data
df = diabetes.drop(columns=['SkinThickness', 'Insulin'])
df = df.dropna() # remove examples with missing data

# X and y data
X = df[df.keys()[0:-1]]
y = df[df.keys()[-1]]

(n,p) = X.shape
print('n =', n, ', p =', p)
```

```
n = 723 , p = 6
```

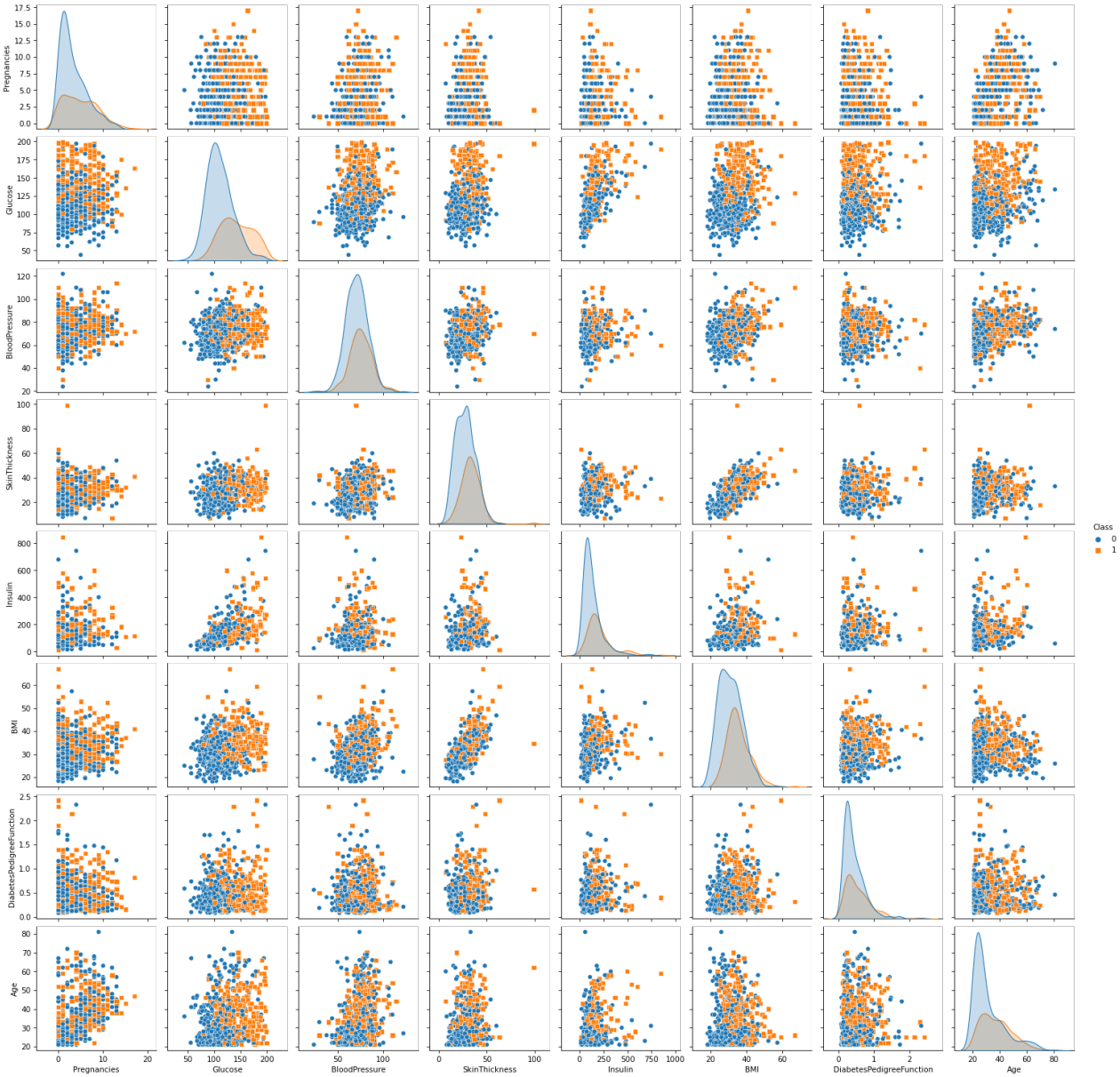
In [5]:
```
# Split training/test set
X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=0.25, random_state=12)
```

(a)

Pairplot

In [6]:
```
plt.figure();
sns.pairplot(data=diabetes,hue = 'Class', markers=["o", "s"])
if svfigs:
    plt.savefig(svdir + 'diabetes1.png', bbox_inches = 'tight');
```

```
<Figure size 450x300 with 0 Axes>
```



(b)

Standardise the data

In [7]:
```
scaler = StandardScaler()
scaled_diabetes = scaler.fit_transform(diabetes)
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.fit_transform(X_test)
```

(c)

LDA

In [8]:
```
# Using Scikit-learn
LDAclassifier = LinearDiscriminantAnalysis()
LDAclassifier.fit(scaled_X_train, y_train)

# Predict the values on the same grid as before
ytest_lda = LDAclassifier.predict(scaled_X_test)# gives the same value as 
# Plot the results (same as before)
diabetestest_lda = pd.concat([scaled_X_test, pd.DataFrame(ytest_lda, colum
plt.figure()
sns.scatterplot(data=diabestest_lda)
if svfigs:
    plt.savefig(svdir + 'diabetesLDAtest2.png', bbox_inches = 'tight')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-4c54053ae163> in <module>
      6 ytest_lda = LDAclassifier.predict(scaled_X_test)# gives the same va
lue as with the above function
      7 # Plot the results (same as before)
----> 8 diabetestest_lda = pd.concat([scaled_X_test, pd.DataFrame(ytest_lda
, columns=['class'])], axis = 1)
      9 plt.figure()
     10 sns.scatterplot(data=diabestest_lda)

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/reshape/concat.py in
concat(objs, axis, join, ignore_index, keys, levels, names, verify_integrit
y, sort, copy)
    272     ValueError: Indexes have overlapping values: ['a']
    273     """
--> 274     op = _Concatenator(
    275         objs,
    276         axis=axis,

/opt/anaconda3/lib/python3.8/site-packages/pandas/core/reshape/concat.py in
__init__(self, objs, axis, join, keys, levels, names, ignore_index, verify_
integrity, copy, sort)
    357                     "only Series and DataFrame objs are valid"
    358                 )
--> 359                 raise TypeError(msg)
    360
    361             # consolidate

TypeError: cannot concatenate object of type '<class 'numpy.ndarray'>'; onl
y Series and DataFrame objs are valid
```

Statistical Machine Learning

Sheet 3 Solutions — HT21

Overfitting, Regularisation, ROC curves, Optimisation, Linear Classifiers

1. The binary Naive Bayes classifier has interesting connections to the logistic regression classifier. You will show that, under certain assumptions, the conditional distribution $\Pr(Y = 1 \mid X = x)$ for Naive Bayes has a similar form to logistic regression. You will then derive the maximum likelihood parameter estimates under these assumptions.

   Suppose $X = (X_1, \ldots, X_p)^\intercal$ is a continuous random vector in $\mathbb{R}^p$, and Y is a binary random variable with values in $\{-1, 1\}$ representing the class labels. Let $\pi_1 = \Pr(Y = 1)$ and g_k denote the conditional probability density function of X given that $Y = k$. We make the Naive Bayes assumption that

$$g_k(x) = \prod_{j=1}^{p} g_{kj}(x_j; \mu_{kj}, \sigma_j^2)$$

   where $x = (x_1, \ldots, x_p)^\intercal \in \mathbb{R}^p$ and

$$g_{kj}(x_j; \mu_{kj}, \sigma_j^2) = (2\pi\sigma_j^2)^{-1/2} e^{-\frac{(x_j - \mu_{kj})^2}{2\sigma_j^2}}.$$

   That is, conditionally on $Y = k$, $X_1, \ldots, X_p$ are independent, and X_j given $Y = k$ follows a normal distribution with mean μ_{kj} and variance σ_j^2. Note that the variance parameter depends on the input dimension j but not on the class k.

   (a) For a given $x = (x_1, \ldots, x_p)^\intercal \in \mathbb{R}^p$, show that

$$\Pr(Y = 1 | X = x) = \frac{1}{1 + \exp(-\beta_0 - \sum_{j=1}^{p} \beta_j x_j)}.$$

   where you should give the explicit form of $\beta_0, \ldots, \beta_p$ in terms of π_1, μ_{kj}, σ_j, for $j = 1, \ldots, p$ and $k \in \{-1, 1\}$.

   Solution:

François Caron: caron@stats.ox.ac.uk

$$\begin{aligned}
\Pr(Y=1|X=x) &= \frac{g_1(x)\Pr(Y=1)}{g_1(x)\Pr(Y=1)+g_{-1}(x)\Pr(Y=-1)} \quad \text{(By Bayes' Rule)} \\
&= \frac{g_1(x)\pi_1}{g_1(x)\pi_1+g_{-1}(x)(1-\pi_1)} \\
&= \frac{1}{1+\frac{g_{-1}(x)(1-\pi_1)}{g_1(x)\pi_1}} \\
&= \frac{1}{1+\exp\big\{-\log\frac{g_1(x)\pi_1}{g_{-1}(x)(1-\pi_1)}\big\}} \\
&= \frac{1}{1+\exp\big\{-\log(\frac{\pi_1}{1-\pi_1})-\log(\frac{g_1(x)}{g_{-1}(x)})\big\}} \\
&= \frac{1}{1+\exp\big\{-\log(\frac{\pi_1}{1-\pi_1})-\sum_{j=1}^p \log(\frac{g_{1,j}(x_j;\mu_{1,j},\sigma_j^2)}{g_{-1,j}(x_j;\mu_{-1,j},\sigma_j^2)})\big\}}.
\end{aligned}$$

Additionally,

$$\begin{aligned}
\log\frac{g_{1,j}(x_j;\mu_{1,j},\sigma_j^2)}{g_{-1,j}(x_j;\mu_{-1,j},\sigma_j^2)} &= \frac{(x_j-\mu_{-1,j})^2-(x_j-\mu_{1,j})^2}{2\sigma_j^2} \\
&= \frac{\mu_{-1,j}^2-\mu_{1,j}^2}{2\sigma_j^2}+\frac{\mu_{1,j}-\mu_{-1,j}}{\sigma_j^2}x_j
\end{aligned}$$

This gives

$$\Pr(Y=1|X=x) = \frac{1}{1+\exp(-\beta_0-\sum_{j=1}^p \beta_j x_j)}.$$

where

$$\begin{aligned}
\beta_0 &= \log\frac{\pi_1}{1-\pi_1}+\sum_{j=1}^p \frac{\mu_{-1,j}^2-\mu_{1,j}^2}{2\sigma_j^2} \\
\beta_j &= \frac{\mu_{1,j}-\mu_{-1,j}}{\sigma_j^2}, \quad \forall j=1,...,p
\end{aligned}$$

(b) This part is independent from part (a).

Suppose a training set with n examples $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ is given, where $x_i=(x_{i1},\ldots,x_{ip})^\top$ is a p-dimensional input vector, and $y_i\in\{-1,1\}$ is its corresponding label. Using the given naive Bayes assumption, provide the maximum likelihood estimate for the parameters $\theta=(\pi_1,(\mu_{1,j},\mu_{-1,j},\sigma_j^2)_{j=1,\ldots,p})$.

Solution: Denote $\pi_{-1}=1-\pi_1$, $m_1=\sum_{i=1}^n \mathbb{1}_{y_i=1}$ the number of positive examples in

the training dataset and $m_{-1} = n - m_1$. The log-likelihood takes the form

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{n} \log(\pi_{y_i} g_{y_i}(x_i)) \\
&= \sum_{i=1}^{n} \log \pi_{y_i} + \sum_{i=1}^{n}\sum_{j=1}^{p} \log g_{y_i,j}(x_{ij}; \mu_{y_i,j}, \sigma_j^2) \\
&= m_1 \log(\pi_1) + (n - m_1)\log(1 - \pi_1) \\
&\quad + \sum_{j=1}^{p} \sum_{i|y_i=1} \log g_{1,j}(x_{ij}; \mu_{1,j}, \sigma_j^2) + \sum_{j=1}^{p} \sum_{i|y_i=-1} \log g_{-1,j}(x_{ij}; \mu_{-1,j}, \sigma_j^2) \\
&= m_1 \log(\pi_1) + (n - m_1)\log(1 - \pi_1) \\
&\quad + \sum_{j=1}^{p} \sum_{i|y_i=1} \left[-\frac{1}{2}\log \sigma_j^2 - \frac{(x_{ij} - \mu_{1,j})^2}{2\sigma_j^2} \right] \\
&\quad + \sum_{j=1}^{p} \sum_{i|y_i=-1} \left[-\frac{1}{2}\log \sigma_j^2 - \frac{(x_{ij} - \mu_{-1,j})^2}{2\sigma_j^2} \right] + \text{const.}
\end{aligned}$$

Differentiating and setting to 0, we obtain

$$\begin{aligned}
\frac{\partial \ell}{\partial \pi_1} &= \frac{m_1}{\pi_1} - \frac{n - m_1}{1 - \pi_1} = 0 \\
\frac{\partial \ell}{\partial \mu_{kj}} &= \sum_{i|y_i=k} \frac{1}{\sigma_j^2}(x_{ij} - \mu_{kj}) = 0 \\
\frac{\partial \ell}{\partial \sigma_j^2} &= -\frac{n}{2\sigma_j^2} + \sum_{i|y_i=1} \left(\frac{1}{2\sigma_j^4}(x_{ij} - \mu_{1,j})^2 \right) + \sum_{i|y_i=-1} \left(\frac{1}{2\sigma_j^4}(x_{ij} - \mu_{-1,j})^2 \right) = 0
\end{aligned}$$

This gives, for $k \in \{-1, 1\}$ and $j = 1, \ldots, p$,

$$\begin{aligned}
\widehat{\pi}_1 &= \frac{m_1}{n} \\
\widehat{\mu}_{kj} &= \frac{1}{m_k} \sum_{i|y_i=k} x_{ij} \\
\widehat{\sigma}_j^2 &= \frac{\sum_{i|y_i=1}(x_{ij} - \widehat{\mu}_{1,j})^2 + \sum_{i|y_i=-1}(x_{ij} - \widehat{\mu}_{-1,j})^2}{n}
\end{aligned}$$

François Caron: caron@stats.ox.ac.uk

2. Assume we trained a classifier using a dataset $d = ((x_1, y_1), \ldots, (x_n, y_n))$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, 2, ..., K\}$. We are interested in classifying a new input vector x_{n+1}. However, we have only been able to collect $p-1$ features, say $(x_{n+1,2}, ..., x_{n+1,p})$ and $x_{n+1,1}$ is missing. Explain whether or not it is possible to use the trained classifier to classify this incomplete input vector in the cases listed below. If it is possible, how do you classify the incomplete test vector?

   Note: You do not need to calculate any integrals in this question. For generative classifiers, $g_k(x)$ denotes the conditional class probability mass/density function of X given $Y = k$.

   (a) A naïve Bayes model, with

$$g_k(x_i) = \prod_{j=1}^{p} g_{kj}(x_{ij}; \theta_{kj}),$$

   i.e. conditioned upon $Y_i = k$, you assume that the different dimensions of the input are independent and X_{ij} given $Y_i = k$ has probability mass/density function $g_{kj}(x_{ij}; \theta_{kj})$.

   Solution: Yes. Since features are assumed independent in naïve Bayes, the vector $(X_{n+1,2}, \ldots, X_{n+1,p})$ conditional on $Y_{n+1} = k$, has the pdf/pmf

$$\prod_{j=2}^{p} g_{kj}(x_{n+1,j}; \theta_{kj})$$

   Using Bayes rule, we obtain

$$\Pr(Y_{n+1} = k | X_{n+1,2} = x_{n+1,2}, \ldots, X_{n+1,p} = x_{n+1,p}) \propto \pi_k \prod_{j=2}^{p} g_{kj}(x_{n+1,j}; \theta_{kj})$$

   i.e. we simply ignore feature 1.

   (b) An LDA model, i.e.

$$g_k(x_i) = \varphi(x_i; \mu_k, \Sigma)$$

   where $\varphi(x; \mu, \Sigma)$ denotes the pdf of the multivariate normal with mean μ and covariance matrix Σ, evaluated at x.

   Solution: Yes. The conditional distribution of the $(p-1)$ dimensional random vector $X_{n+1,2}, \ldots, X_{n+1,p}$ given $Y = k$ is multivariate normal with mean $\mu'_k = (\mu_{k2}, \ldots, \mu_{kp})^\top$ and covariance matrix Σ' obtained from Σ by dropping the first row and the first column. Using Bayes rule, we obtain

$$\Pr(Y_{n+1} = k | X_{n+1,2} = x_{n+1,2}, \ldots, X_{n+1,p} = x_{n+1,p}) \propto \pi_k \varphi\left(x'_{n+1}; \mu'_k, \Sigma'\right)$$

   where $x'_{n+1} = (x_{n+1,2}, \ldots, x_{n+1,p})^\intercal$.

(c) Generally, what condition on the conditional pdf/pmf $g_k(x)$ would allow easy classification (i.e, without numerical integration) in the presence of missing features for generative classifiers like LDA or naïve Bayes?

Solution: For a vector x_i, and $A \subseteq \{1, \ldots, p\}$ corresponding to the set of indices of the dimensions of the input vector that are missing, let $x_{iA} = (x_{ij} | j \in A)$. We need to be able to compute analytically the marginals of the class-conditional distributions. For example, assuming a continuous input $x_i \in \mathbb{R}^p$, we need to be able to compute analytically

$$\int_{\mathbb{R}^{|A|}} g_k(x_i) dx_{iA}$$

for any A.

(d) A binary logistic regression model

$$\Pr(Y = 1 | X = x) = \text{sig}(a + b^\intercal x).$$

Solution: It would not directly be possible, since logistic regression does not model the full joint distribution so does not give predictions when the input data vector is partially observed. There are various other ways to get around this - data imputation etc.

François Caron: caron@stats.ox.ac.uk

3. Let $d_{\text{train}} = ((x_1, y_1), \ldots, (x_n, y_n))$ and $d_{\text{test}} = ((\widetilde{x}_1, \widetilde{y}_1), \ldots, (\widetilde{x}_m, \widetilde{y}_m))$ be respectively the training dataset and test dataset. These are realisations of the random samples D_{train} and D_{test}, which are composed of iid random variables with the same distribution as (X, Y). Let L be a loss function, and denote $R(h) = \mathbb{E}[L(Y, h(X))]$ be the associated population risk. Denote

$$\widehat{R}_n^{(d_{\text{train}})}(h) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))$$

be the empirical risk of a prediction rule h on the training set. Similarly, let

$$\widehat{R}_m^{(d_{\text{test}})}(h) = \frac{1}{m} \sum_{i=1}^{m} L(\widetilde{y}_i, h(\widetilde{x}_i))$$

be the empirical risk of a prediction rule h on the test set. Let $\mathcal{H}$ be some class of prediction rules. Let

$$\widehat{h}^{(d_{\text{train}})} = \arg\min_{h \in \mathcal{H}} \widehat{R}_n^{(d_{\text{train}})}(h)$$

be the empirical risk minimiser and

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} R(h)$$

the best prediction rule in the class $\mathcal{H}$.

Show that

$$\mathbb{E}[\widehat{R}_n^{(D_{\text{train}})}(\widehat{h}^{(D_{\text{train}})})] \leq R(h^*_{\mathcal{H}}) \leq \mathbb{E}[\widehat{R}_m^{(D_{\text{test}})}(\widehat{h}^{(D_{\text{train}})})]$$

where the expectations are taken with respect to the training sample D_{train} and test sample D_{test}.

Solution: By definition of the empirical risk minimiser, for any $h \in \mathcal{H}$, and any training set d_{train}

$$\widehat{R}_n^{(d_{\text{train}})}(\widehat{h}^{(d_{\text{train}})}) \leq \widehat{R}_n^{(d_{\text{train}})}(h)$$

Taking expectation with respect to D_{train}, and using the fact that for any fixed prediction rule h, $\mathbb{E}\left[\widehat{R}_n^{(D_{\text{train}})}(h)\right] = R(h)$, we obtain

$$\mathbb{E}[\widehat{R}_n^{(D_{\text{train}})}(\widehat{h}^{(D_{\text{train}})})] \leq R(h)$$

for any $h \in \mathcal{H}$, and in particular for $h^*_{\mathcal{H}}$. Similarly

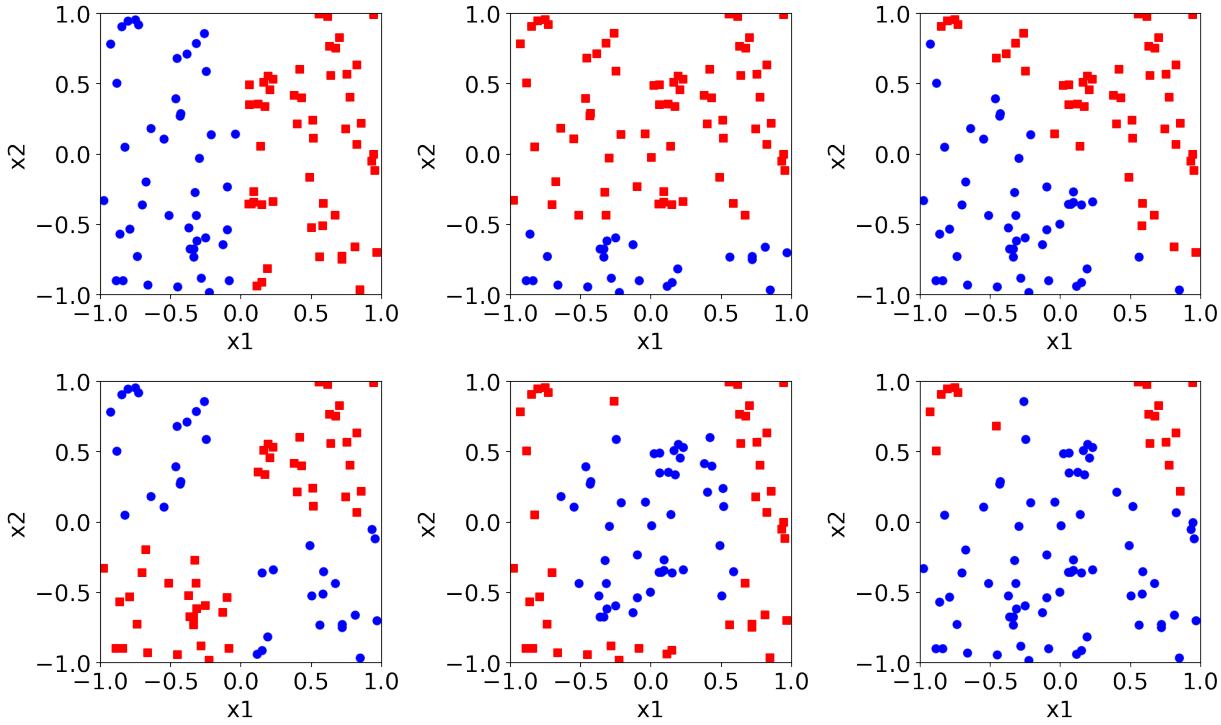
$$\begin{aligned}\mathbb{E}[\widehat{R}_m^{(D_{\text{test}})}(\widehat{h}^{(D_{\text{train}})})] &= \mathbb{E}\left[\mathbb{E}\left[\widehat{R}_m^{(D_{\text{test}})}(\widehat{h}^{(D_{\text{train}})}) \mid D_{\text{train}}\right]\right] \\ &= \mathbb{E}\left[R(\widehat{h}^{(D_{\text{train}})})\right]\end{aligned}$$

By definition of $h^*_{\mathcal{H}}$, $R(h^*_{\mathcal{H}}) \leq R(h)$ for any prediction rule $h \in \mathcal{H}$. Hence $\mathbb{E}\left[R(\widehat{h}^{(D_{\text{train}})})\right] \geq \mathbb{E}\left[R(h^*_{\mathcal{H}})\right] = R(h^*_{\mathcal{H}})$.

This shows that the training risk $\widehat{R}_n^{(D_{\text{train}})}(\widehat{h}^{(D_{\text{train}})})$ underestimates the population risk $R(\widehat{h}^{(D_{\text{train}})})$ of the learned prediction rule. With probability one, $R(\widehat{h}^{(D_{\text{train}})}) \geq R(h^*_{\mathcal{H}})$, but

$$\mathbb{E}[\widehat{R}_n^{(D_{\text{train}})}(\widehat{h}^{(D_{\text{train}})})] \leq R(h^*_{\mathcal{H}}).$$

4. For each of the datasets below, find a potentially non-linear function $\phi : \mathbb{R}^2 \to \mathbb{R}^M$ which makes the transformed inputs linearly separable (limit yourself to polynomial expansions with interaction terms). That is, there exists a parameter β such that the discriminant function $f(x) = \phi(x)^\intercal \beta$ perfectly separates the data.



Solution: From left to right and top to bottom:

1. Decision boundary $x_1 = 0$, so could use $\phi(x) = x_1$
2. Decision boundary $x_2 = -.5$, so could use $\phi(x) = (1 \;\; x_2)^\intercal$
3. Decision boundary $x_2 = -x_1$, so could use $\phi(x) = (x_1 \;\; x_2)^\intercal$
4. Decision boundary $x_1 x_2 = 0$ ($x_1 = 0$ or $x_2 = 0$), so could use $\phi(x) = (x_1 x_2)^\intercal$
5. Decision boundary is an ellipse (centered at 0) $ax_1^2 + bx_2^2 = c$, so could use $\phi(x) = (1 \;\; x_1^2 \;\; x_2^2)^\intercal$
6. Decision boundary of the form $x_2 = a/x_1^2$, could use $\phi(x) = (1 \;\; x_1^2 x_2)^\intercal$

5. (a) Show that if g_1 and g_2 are convex functions on $\mathbb{R}^p$ and c_1, c_2 are nonnegative scalars, $c_1 g_1 + c_2 g_2$ is convex.

   (b) Show that if $g : \mathbb{R} \to \mathbb{R}$ is convex, $\psi(\beta) = g(a^\intercal \beta + b)$ is convex where a, β are p-dimensional vectors and b is a scalar.

   (c) Show that the following functions are convex: $g_1(x) = |x|$, $g_2(x) = e^{-x}$, $g_3(x) = (1-x)^2$, $g_4(x) = \log(1+e^{-x})$, $g_5(x) = \max(0, -x)$.

   (d) What can you conclude about the objective functions of the least-squares, perceptron and logistic regression (linear) classifiers, with a L2 regularisation?

Solution:

Let $\alpha \in [0,1]$.

$$\begin{aligned} &c_1 g_1(\alpha x + (1-\alpha)y) + c_2 g_2(\alpha x + (1-\alpha)y) \\ &\quad \leq c_1(\alpha g_1(x) + (1-\alpha) g_1(y)) + c_2(\alpha g_2(x) + (1-\alpha) g_2(y)) \\ &\quad = \alpha(c_1 g_1(x) + c_2 g_2(x)) + (1-\alpha)(c_1 g_1(y) + c_2 g_2(y)) \end{aligned}$$

$$\begin{aligned} \psi(\alpha\beta + (1-\alpha)\gamma) &= g(a^\intercal(\alpha\beta + (1-\alpha)\gamma) + b) \\ &= g(\alpha(a^\intercal\beta + b) + (1-\alpha)(a^\intercal\gamma + b)) \\ &\leq \alpha g(a^\intercal\beta + b) + (1-\alpha) g(a^\intercal\gamma + b) \end{aligned}$$

Using the triangle inequality

$$|\alpha x + (1-\alpha)y| \leq \alpha|x| + (1-\alpha)|y|$$

g_2, g_3 and g_4 are twice differentiable

$$\begin{aligned} g_2''(x) &= e^{-x} > 0 \\ g_3''(x) &= 2 > 0 \\ g_4''(x) &= \frac{e^{-x}}{(1+e^{-x})^2} > 0. \end{aligned}$$

We have

$$g_5(x) = \max(0, -x) = \frac{1}{2}(-x) + \frac{1}{2}|x|$$

so it is a nonnegative linear combination of convex functions.

For L2-regularised least-squares, perceptron and logistic regression, the objective function is of the form

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \psi(y_i \phi(x_i)^\intercal \beta) + \frac{\lambda}{n}\sum_{j=1}^{p} \beta_j^2$$

where the function ψ is convex. By the previous parts, J is a nonnegative linear combination of convex functions, and is therefore convex.

6. Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \mathbb{R}$ the target space. Consider the class of linear prediction rules $h(x) = \phi(x)^\intercal \beta$ with input expansion $\phi : \mathcal{X} \to \mathbb{R}^p$ and $\beta \in \mathbb{R}^p$. The regularised empirical risk minimiser, with L2 penalty, is $\widehat{h}(x) = \phi(x)^\intercal \widehat{\beta}$ where $\widehat{\beta}$ minimises the ridge regression objective function

$$\begin{aligned} J(\beta) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \phi(x_i)^\intercal\beta)^2 + \frac{\lambda}{n}\|\beta\|^2 \\ &= \frac{1}{n}\|\,\mathbf{y} - \mathbf{\Phi}\beta\|^2 + \frac{\lambda}{n}\|\beta\|^2 \end{aligned} \tag{1}$$

where $\lambda > 0$ is the regularisation parameter.

(a) Show that $\widehat{\beta}$ can be interpreted as the maximum a posteriori (that is, the maximum of the posterior probability density function) of the parameter β under the model

$$\mathbf{y} \mid \beta \sim \mathcal{N}(\mathbf{\Phi}\beta, \sigma^2 I_n), \qquad \beta \sim \mathcal{N}(0, \tau I_p)$$

where $\tau > 0$, $\sigma > 0$, and I_k denotes the k-by-k identity matrix. Find the relationship between the regularisation parameter λ and the variances τ and σ^2.

Solution: The posterior pdf is

$$\begin{aligned} p(\beta \mid \mathbf{y}) &\propto p(\mathbf{y} \mid \beta)p(\beta) \\ &\propto \exp\left(-\frac{\|\,\mathbf{y} - \mathbf{\Phi}\beta\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|\beta\|^2}{2\tau}\right) \end{aligned}$$

where the term independent of β are ignored. The maximum a posteriori is

$$\begin{aligned} \widehat{\beta} &= \arg\max_{\beta} p(\beta \mid \mathbf{y}) \\ &= \arg\max_{\beta} \log p(\beta \mid \mathbf{y}) \\ &= \arg\max_{\beta} -\frac{\|\,\mathbf{y} - \mathbf{\Phi}\beta\|^2}{2\sigma^2} - \frac{\|\beta\|^2}{2\tau} \\ &= \arg\min_{\beta} \frac{\|\,\mathbf{y} - \mathbf{\Phi}\beta\|^2}{n} + \frac{\sigma^2}{n\tau}\|\beta\|^2 \end{aligned}$$

hence it corresponds to the ridge regression estimator with regularisation parameter $\lambda = \frac{\sigma^2}{\tau}$.

(b) Show that $\widehat{\beta}$ can also be obtained by applying ordinary least squares regression (that is, without regularisation) on an augmented dataset $\widetilde{\mathbf{y}}, \widetilde{\mathbf{\Phi}}$, that depends on $\mathbf{y}$, $\mathbf{\Phi}$ and λ.

Solution:

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \phi(x_i)^\intercal\beta)^2 + \frac{1}{n}\sum_{j=1}^{p}(\sqrt{\lambda}\beta_j)^2$$

So setting $\widetilde{\mathbf{y}} = (y_1, \ldots, y_n, 0, \ldots, 0)^\intercal \in \mathbb{R}^{n+p}$ and $\widetilde{\mathbf{\Phi}}$ is the $(n+p)-$by$-p$ matrix defined by

$$\widetilde{\mathbf{\Phi}} = \begin{pmatrix} \mathbf{\Phi} \\ \sqrt{\lambda} I_p \end{pmatrix}$$

we obtain

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\|\widetilde{\mathbf{y}} - \widetilde{\mathbf{\Phi}}\beta\|^2.$$

(c) Give the gradient descent algorithm to minimise (1).

Solution:

$$\nabla_\beta J(\beta) = \frac{1}{n}(2\mathbf{\Phi}^\intercal\mathbf{\Phi}\beta - 2\mathbf{\Phi}^\intercal\mathbf{y} + 2\lambda\beta)$$

This gives the update

$$\beta^{(t+1)} = \beta^{(t)} - \frac{2\eta}{n}((\mathbf{\Phi}^\intercal\mathbf{\Phi} + \lambda I_p)\beta^{(t)} - \mathbf{\Phi}^\intercal\mathbf{y})$$

7. We consider a dataset originally collected by the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset consists of several medical predictors (Blood pressure, body mass index, age, etc) and a binary output, indicating if the person has diabete or not. All patients here are female, at least 21 years old and of Pima Indian heritage. The objective is to derive a classifier for predicting if a given person is diabetic or not, based on its medical inputs. The dataset can be downloaded from `https://www.kaggle.com/kumargh/pimaindiansdiabetescsv`.

   The following code loads the dataset and removes missing data. The cleaned dataset has $n = 724$ patients with $p = 6$ inputs. The data is split between a training and a test set.

```
# Standard libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
# Train-test and preprocessing functions
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Classifiers
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
# Metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

# Download the data from
# https://www.kaggle.com/uciml/pima-indians-diabetes-database
filename = '../datasets/pimaIndians/diabetes.csv'
diabetes = pd.read_csv(filename)

# Missing values indicated by 0 - replace by nan
for name in diabetes.keys()[1:-1]:
# for features other than pregnancies, 0 indicate a missing value, so replace by nan
    diabetes[name] = diabetes[name].replace({0: np.nan})
diabetes.isna().sum()

 # remove the dimensions with large number of missing data
df = diabetes.drop(columns=['SkinThickness', 'Insulin'])
df = df.dropna() # remove examples with missing data
# X and y data
X = df[df.keys()[0:-1]]
y = df[df.keys()[-1]]
```


François Caron: caron@stats.ox.ac.uk

```
(n,p) = X.shape
print('n =', n, ', p =', p)

# Split training/test set
X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=0.25, random_state=12)
```

(a) Visualise the data using a pairplot

(b) Standardise the training data, and apply the same transformation to the test data (use the function `StandardScaler`)

(c) For each of the following plug-in classifiers: Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naive Bayes, Logistic Regression:

- Estimate the parameters of the classifiers on the training data
- Under the 0-1 loss, report the accuracy and confusion matrix on the training and test datasets
- Plot the ROC curve and calculate the Area Under the Curve (AUC) on both the training and test set

(d) (Optional): Try to apply some input transformation, adding interaction terms. Does it improve the performances in terms of AUC?

Solution: See Python Notebook

François Caron: caron@stats.ox.ac.uk

Statistical Machine Learning

Sheet 4 — HT21

Nearest neighbours, neural networks, Random Forests, classification trees

1. (Classification Trees) We are deciding how to split a node in a binary classification tree. The node contains 300 data vectors of class 1 and 100 of class 2. Write this as $(300, 100)$. We can split the node either as (a) $(200, 100)$, $(100, 0)$, or (b) $(150, 50)$, $(150, 50)$. Compute the change in the impurity measure when using the misclassification rate and when using the Gini impurity for two splits. Which split is preferred?

2. (Neural networks) Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$ (note that we use a 0-1 coding here). Consider a dataset $d = (x_i, y_i)_{i=1,\dots,n}$. We consider a one-hidden layer neural network with m hidden units and a sigmoid activation function. The objective function, using the L_2-regularised log-loss, is:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \eta_\theta(x_i) + (1-y_i)\log(1-\eta_\theta(x_i))\right] + \frac{\lambda}{n}\left(\sum_{jl}(w_{jl}^{(h)})^2 + \sum_{l}(w_l^{(o)})^2\right)$$

where, for $i = 1, \dots, n$ and $k = 1, \dots, m$,

$$\eta_\theta(x_i) = \text{sig}\left(b^{(o)} + \sum_{k=1}^{m} w_k^{(o)} z_{ik}\right), \qquad z_{ik} = \text{sig}\left(b_k^{(h)} + \sum_{j=1}^{p} w_{jk}^{(h)} x_{ij}\right),$$

where $\text{sig}(a) = \frac{1}{1+e^{-a}}$ is the sigmoid activation function.

   (a) Verify that the derivatives needed for gradient descent are:

$$\frac{\partial J}{\partial w_k^{(o)}} = \frac{1}{n}\left(2\lambda w_k^{(o)} + \sum_{i=1}^{n}(\eta_\theta(x_i) - y_i) z_{ik}\right),$$
$$\frac{\partial J}{\partial w_{jk}^{(h)}} = \frac{1}{n}\left(2\lambda w_{jk}^{(h)} + \sum_{i=1}^{n}(\eta_\theta(x_i) - y_i) w_k^{(o)} z_{ik}(1 - z_{ik}) x_{ij}\right).$$

   (b) Suppose instead that you have a neural network for binary classification with $\ell \geq 2$ hidden layers, each hidden layer having m neurons with sigmoid activation function. Give the parameterisation for each layer. Considering empirical risk minimisation under a surrogate log-loss, with no penalisation, derive the backpropagation algorithm to compute the derivatives of the objective function with respect to the parameters. For simplicity, you can ignore the bias terms.

François Caron: caron@stats.ox.ac.uk

3. (Asymptotic properties of 1-nearest neighbour) Let $D_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$ be a random sample of size n, iid with the same distribution as (X, Y) where $X \in \mathbb{R}$ and $Y \in \{-1, 1\}$. Assume that X is a continuous random variable with pdf f_X such that $f_X(x) > 0$ for all $x \in \mathbb{R}$. Let $\eta(x) = \Pr(Y = 1 \mid X = x)$ and assume η is a continuous function.

   (a) Let h^* be the Bayes classifier under the 0-1 loss. Show that the conditional Bayes risk under the 0-1 loss is given by

   $$R_x(h^*) := \mathbb{E}[\mathbb{1}_{Y \neq h^*(X)} \mid X = x] = \Pr(Y \neq h^*(X) \mid X = x) = \min(\eta(x), 1 - \eta(x)).$$

   (b) For any test point $\widetilde{x} \in \mathbb{R}$, denote $X_{n,(1)} \in \mathbb{R}$ the nearest neighbour of $\widetilde{x}$ in the set $\{X_1, \ldots, X_n\}$, and $Y_{n,(1)}$ its corresponding class. The 1-nearest neighbour (1-NN) classifier is defined as

   $$\widehat{h}^{(D_n)}(\widetilde{x}) = Y_{n,(1)}.$$

   Let $(\widetilde{X}, \widetilde{Y})$ be a test sample, with the same distribution as (X, Y), and independent of the training sample D_n. Show that

   $$\Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}, X_{n,(1)} = x_{n,(1)}) = \eta(\widetilde{x})(1 - \eta(x_{n,(1)})) + (1 - \eta(\widetilde{x}))\eta(x_{n,(1)}).$$

   (c) Show that, for any $\widetilde{x} \in \mathbb{R}$, $X_{n,(1)}$ converges to $\widetilde{x}$ almost surely as $n \to \infty$.

   [Hint: first prove that the convergence holds in probability. Then use the following lemma:
   Let $Z_1 \geq Z_2 \geq Z_3 \geq \ldots$ be a monotone decreasing sequence of random variables. If $Z_n \to Z$ in probability, then $Z_n \to Z$ almost surely.]

   (d) Give the limit, as $n \to \infty$, of

   $$\Pr\left(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}\right).$$

   Deduce that

   $$R(h^*) \leq \lim_{n \to \infty} \mathbb{E}[R(\widehat{h}^{(D_n)})] \leq 2R(h^*)(1 - R(h^*))$$

   where, for a classifier h, $R(h) = \mathbb{E}[\mathbb{1}_{Y \neq h(X)}] = \Pr(Y \neq h(X))$ is the risk under the 0-1 loss.

   [Hint: you may use this simplified version of the dominated convergence theorem (see Part A Integration for details).
   *Let $(X_1, X_2, \ldots)$ be a **bounded** sequence of random variables such that $X_n \to X$ almost surely. Then $\lim_{n \to \infty} \mathbb{E}[X_n] = \mathbb{E}[X]$.]*

François Caron: caron@stats.ox.ac.uk

4. (Surrogate loss functions) Consider a binary classification problem, with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{-1, 1\}$. Let h be a classifier

$$h(x) = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

with discriminant function $f : \mathcal{X} \to \mathbb{R}$ taking values in $\mathcal{F}_f$, the set of functions from $\mathcal{X}$ to $\mathbb{R}$. For a surrogate loss function ψ, let

$$R_\psi(f) = \mathbb{E}[\psi(Yf(X))]$$

be the ψ-risk of the discriminant function f. If $\psi(z) = \mathbb{1}_{z<0}$, then $R_\psi(f) = \Pr(Yf(X) < 1) = \Pr(Y \neq \text{sign}(f(X)))$ is the usual risk under the 0-1 loss. Denote

$$h^*_\psi = \begin{cases} 1 & \text{if } f^*_\psi(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

with

$$f^*_\psi = \arg\min_{f \in \mathcal{F}_f} R_\psi(f)$$

the (optimal) Bayes classifier under the surrogate ψ loss. Denote h^* the Bayes classifier under the 0-1 loss.

(a) For each of the following surrogate functions, find the associated (optimal) discriminant function f^*_ψ and show that $h^*_\psi = h^*$

$$\psi_1(z) = e^{-z}$$
$$\psi_2(z) = (1 - z)^2$$
$$\psi_3(z) = \log(1 + e^{-z})/\log(2)$$

(b) Let $\mathcal{X} = \mathbb{R}$. Consider the class $\mathcal{H}_f \subset \mathcal{F}_f$ of linear discriminant functions of the form $f(x) = \beta_0 + \beta_1 x$. Denote

$$f^*_{\psi,\mathcal{H}_f} = \arg\min_{f \in \mathcal{H}_f} R_\psi(f)$$

the optimal discriminant function **amongst this class** for the surrogate risk ψ, and let $h^*_{\psi,\mathcal{H}_f}$ be the associated classifier. Denote $f^*_{\mathcal{H}_f}$ the optimal discriminant function in this class under the 0-1 loss and $h^*_{\mathcal{H}_f}$ the associated classifier.

Assuming that $\Pr(Y = 1) = \Pr(Y = -1) = 1/2$, Show that, for the surrogate loss ψ_2, we have $f^*_{\psi_2,\mathcal{H}_f}(\mathbb{E}(X)) = 0$. Deduce that in general, $h^*_{\psi_2,\mathcal{H}_f} \neq h^*_{\mathcal{H}_f}$.

François Caron: caron@stats.ox.ac.uk

5. (Coding: Classification trees and Random Forests)

   The code below loads the wine dataset, that we have already analysed with Linear Discriminant Analysis in Problem Sheet 2.

   (a) We first consider a classification tree.

   (i) Fit a full classification tree to the data and print the tree. Relate the classification rule discovered there to the parameters estimated by LDA.

   (ii) For trees with different number of leaf nodes, report the risk, approximated using 5-fold cross-validation (use the function `cross_val_score`).

   (iii) For the number of leaf nodes that minimises the cross-validation risk, refit the tree on the whole dataset, and plot the tree.

   (b) We now consider a random forest classifier.

   (i) For 500 trees, and different number of features $m = 1, \ldots, 6$ considered for each tree (option max_features in the function `RandomForestClassifier`), calculate and plot the out-of-bag misclassification error.

   (ii) For 500 trees and $m = 3$ features used per tree, plot the variable importances.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler # to normalise the data
from sklearn.datasets import load_wine
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
plt.rcParams['figure.dpi'] = 200

# Load wine data
(X,y) = load_wine(return_X_y = True, as_frame = True)
Xy = pd.concat([X, y], axis=1)

(n, p) = X.shape # number of examples and dimension of each example
K = y.unique().size # number of classes
print('n=', n, ', p=', p, ', K=', K)
X.head()
```

6. (Coding: Neural Networks)

   In this question, we will investigate the use of shallow and multi-layer neural networks for multiclass classification. The code below loads the digits dataset, that we considered in Problem Sheet 2. Recall that the objective is to classify 64×64 images of handwritten digits from 0 to 9. The provided code splits the dataset into a training and test set, and standardises the data.

   (a) Fit a logistic regression model on the training data, and report the accuracy on the training and test sets.

   (b) We now consider a shallow neural network (one hidden layer) with $m = 50$ neurons and tanh activation function. For different values $n_b = 1, 32, 100, 898$ of the batch size (parameter `batch_size` in the function `MLPClassifier`), report the value of the objective function as a function of the number of epochs (where one epoch corresponds to a pass through the entire dataset), when using a stochastic gradient algorithm (you should set `solver='SGD'` in the `MLPClassifier` function).

   (c) Consider now changing the structure of the shallow neural network, that is the number m of neurons. For $m = 5, 10, 20, 100, 200, 300$, fit the neural network parameters. Report the accuracy on the test and training sets as a function of the number of neurons m.

   (d) Consider now a feedforward neural network, with the following number of neurons at the different hidden layers: (100,100,10). Fit the parameters of this model, and report the accuracy on the test and training data.

   (e) (Optional) Try to change the structure of the network, the activation function, or the parameters of the optimiser (e.g. `max_iter, tol, n_iter_no_change`) to improve the accuracy.

```
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler # to normalise the data

# Load data
(X, y) = load_digits(return_X_y=True, as_frame=True, n_class = 10)
```

```
# Plot some images in the dataset
for i in range(30):
    plt.subplot(3, 10, i + 1)
    image = X.loc[i,:].to_numpy()
    plt.imshow(image.reshape(8, 8), cmap=plt.cm.gray)
    plt.title(y[i])
    plt.xticks(())
    plt.yticks(())

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
print('Training:', X_train.shape)
print('Test:', X_test.shape)

# Scale
scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)
```

7. (Optional. Coding: Convolutional Neural Networks) The attached Python notebook implements a convolutional neural network for classifying 28-by-28 images of handwritten digits from the MNIST dataset, a popular machine learning benchmark for multiclass classification. It uses TensorFlow, one of a number of popular deep learning packages.

   (a) If needed, install Tensorflow; run the code.

   (b) Change the parameters of the convolutional neural networks (e.g. number, size of the filter, number of convolutional layers, etc.), and report the accuracy on the training/test data.

François Caron: caron@stats.ox.ac.uk

Statistical Machine Learning

Sheet 4 Solutions — HT21

Nearest neighbours, neural networks, Random Forests, classification trees

1. (Classification Trees) We are deciding how to split a node in a binary classification tree. The node contains 300 data vectors of class 1 and 100 of class 2. Write this as $(300, 100)$. We can split the node either as (a) $(200, 100)$, $(100, 0)$, or (b) $(150, 50)$, $(150, 50)$. Compute the change in the impurity measure when using the misclassification rate and when using the Gini impurity for two splits. Which split is preferred?

   Solution: Denote by $\eta_1 = 3/4$ the fraction of vectors labelled 1 at the current node. The change in impurity is given by

   $$\Delta = i(\eta_1) - q_{\text{left}} i(\eta_{\text{left},1}) - q_{\text{right}} i(\eta_{\text{right},1}),$$

   where q_{left} is the fraction of vectors directed left, $\hat{\eta}_{\text{left},1}$ is the fraction of vectors labelled 1 in the left node after the split, and i is the impurity measure. For misclassification rate, $i(\eta) = \min(\eta, 1-\eta)$ and for Gini impurity $i(\eta) = 2\eta(1-\eta)$. Thus for misclassification rate:

   $$\begin{aligned} \Delta_a &= 1/4 - \left(\frac{300}{400} \cdot \frac{100}{300} + \frac{100}{400} \cdot \frac{0}{100} \right) = 0, \\ \Delta_b &= 1/4 - \left(\frac{200}{400} \cdot \frac{50}{200} + \frac{200}{400} \cdot \frac{50}{200} \right) = 0. \end{aligned}$$

   Misclassification rate views two splits as equally pure.

   For Gini impurity

   $$\begin{aligned} \Delta_a &= 2 \cdot \frac{3}{4} \cdot \frac{1}{4} - \left(\frac{3}{4} \cdot 2 \cdot \frac{2}{3} \cdot \frac{1}{3} + \frac{1}{4} \cdot 2 \cdot 1 \cdot 0 \right) = 1/24, \\ \Delta_b &= 2 \cdot \frac{3}{4} \cdot \frac{1}{4} - \left(\frac{1}{2} \cdot 2 \cdot \frac{3}{4} \cdot \frac{1}{4} + \frac{1}{2} \cdot 2 \cdot \frac{3}{4} \cdot \frac{1}{4} \right) = 0, \end{aligned}$$

   so it favours (a).

François Caron: caron@stats.ox.ac.uk

2. (Neural networks) Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$ (note that we use a 0-1 coding here). Consider a dataset $d = (x_i, y_i)_{i=1,\dots,n}$. We consider a one-hidden layer neural network with m hidden units and a sigmoid activation function. The objective function, using the L_2-regularised log-loss, is:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \eta_\theta(x_i) + (1-y_i)\log(1-\eta_\theta(x_i))\right] + \frac{\lambda}{n}\left(\sum_{jl}(w_{jl}^{(h)})^2 + \sum_{l}(w_l^{(o)})^2\right)$$

where, for $i = 1, \dots, n$ and $k = 1, \dots, m$,

$$\eta_\theta(x_i) = \text{sig}\left(b^{(o)} + \sum_{k=1}^{m} w_k^{(o)} z_{ik}\right), \qquad z_{ik} = \text{sig}\left(b_k^{(h)} + \sum_{j=1}^{p} w_{jk}^{(h)} x_{ij}\right),$$

where $\text{sig}(a) = \frac{1}{1+e^{-a}}$ is the sigmoid activation function.

(a) Verify that the derivatives needed for gradient descent are:

$$\frac{\partial J}{\partial w_k^{(o)}} = \frac{1}{n}\left(2\lambda w_k^{(o)} + \sum_{i=1}^{n}(\eta_\theta(x_i) - y_i) z_{ik}\right),$$
$$\frac{\partial J}{\partial w_{jk}^{(h)}} = \frac{1}{n}\left(2\lambda w_{jk}^{(h)} + \sum_{i=1}^{n}(\eta_\theta(x_i) - y_i) w_k^{(o)} z_{ik}(1-z_{ik}) x_{ij}\right).$$

Solution: The first terms are the derivatives of the regularization terms. The second terms are derivatives of the empirical risk terms, for which we will use the chain rule. Recall that

$$\frac{d\,\text{sig}(x)}{dx} = \text{sig}(x)(1-\text{sig}(x)).$$

$$\begin{aligned}
n\frac{\partial J}{\partial w_k^{(o)}} &= 2\lambda w_k^{(o)} + n\sum_{i=1}^{n}\frac{\partial J}{\partial \eta_\theta(x_i)}\frac{\partial \eta_\theta(x_i)}{\partial w_k^{(o)}} \\
&= 2\lambda w_k^{(o)} + \sum_{i=1}^{n}\left(-\frac{y_i}{\eta_\theta(x_i)} + \frac{(1-y_i)}{1-\eta_\theta(x_i)}\right)\eta_\theta(x_i)(1-\eta_\theta(x_i)) z_{ik} \\
&= 2\lambda w_k^{(o)} + \sum_{i=1}^{n}(-y_i(1-\eta_\theta(x_i)) + (1-y_i)\eta_\theta(x_i)) z_{ik} \\
&= 2\lambda w_k^{(o)} + \sum_{i=1}^{n}(\eta_\theta(x_i) - y_i) z_{ik}
\end{aligned}$$

$$
\begin{aligned}
n\frac{\partial J}{\partial w_{jk}^{(h)}} &= 2\lambda w_{jk}^{(h)} + n\sum_{i=1}^{n}\frac{\partial J}{\partial \eta_\theta(x_i)}\frac{\partial \eta_\theta(x_i)}{\partial z_{ik}}\frac{\partial z_{ik}}{\partial w_{jk}^{(h)}} \\
&= 2\lambda w_{jk}^{(h)} + \sum_{i=1}^{n}\left(-\frac{y_i}{\eta_\theta(x_i)} + \frac{1-y_i}{1-\eta_\theta(x_i)}\right)\left(\eta_\theta(x_i)(1-\eta_\theta(x_i))w_k^{(o)}\right)(z_{ik}(1-z_{ik})x_{ij}) \\
&= 2\lambda w_{jk}^{(h)} + \sum_{i=1}^{n}(\eta_\theta(x_i)-y_i)w_k^{(o)}z_{ik}(1-z_{ik})x_{ij}
\end{aligned}
$$

(b) Suppose instead that you have a neural network for binary classification with $\ell \geq 2$ hidden layers, each hidden layer having m neurons with sigmoid activation function. Give the parameterisation for each layer. Considering empirical risk minimisation under a surrogate log-loss, with no penalisation, derive the backpropagation algorithm to compute the derivatives of the objective function with respect to the parameters. For simplicity, you can ignore the bias terms.

Solution: The network is defined as follows. For simplicity, let $z_i^{(0)} = x_i \in \mathbb{R}^p$ be the input vector, and $z_i^{(l)} \in \mathbb{R}^m$ be the vector of latent representation of the example x_i at the lth hidden layer, for $l = 1, \ldots, \ell$. Finally let $z_i^{(\ell+1)} = \eta_\theta(x_i)$ be the output probability. We have:

$$z_i^{(l+1)} = \underline{s}(W^{(l+1)\top} z_i^{(l)}),$$

where $\underline{s}$ is the entry-wise sigmoid function (returns a vector with sigmoid function applied at each element), the parameters are $W^{(l+1)}$, a matrix of weights (of appropriate sizes) for each l, with entries $w_{jk}^{(l+1)}$. Under the surrogate log-loss function, the objective function is:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log z_i^{(\ell+1)} + (1-y_i)\log(1-z_i^{(\ell+1)}).$$

We will compute the partial derivatives with respect to the unit activations z_{ik}^l recursively, starting from layer $\ell+1$ and working backwards towards layer 0:

$$
\begin{aligned}
n\frac{\partial J}{\partial z_i^{(\ell+1)}} &= -\frac{y_i}{z_i^{(\ell+1)}} + \frac{1-y_i}{1-z_i^{(\ell+1)}} \\
\frac{\partial J}{\partial z_{ij}^{(l)}} &= \sum_k \frac{\partial J}{\partial z_{ik}^{(l+1)}}\frac{\partial z_{ik}^{(l+1)}}{\partial z_{ij}^{(l)}} \\
&= \sum_k \frac{\partial J}{\partial z_{ik}^{(l+1)}} z_{ik}^{(l+1)}(1-z_{ik}^{(l+1)})w_{jk}^{(l+1)}
\end{aligned}
$$

Finally, we can compute the partial derivatives with respect to the parameters:

$$\begin{aligned}\frac{\partial J}{\partial w_{jk}^{(l)}} &= \sum_{i=1}^{n} \frac{\partial J}{\partial z_{ik}^{(l)}} \frac{\partial z_{ik}^{(l)}}{\partial w_{jk}^{(l)}} \\ &= \sum_{i=1}^{n} \frac{\partial J}{\partial z_{ik}^{(l)}} z_{ik}^{(l)}(1 - z_{ik}^{(l)}) z_{ij}^{(l-1)}\end{aligned}$$

3. (Asymptotic properties of 1-nearest neighbour) Let $D_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$ be a random sample of size n, iid with the same distribution as (X, Y) where $X \in \mathbb{R}$ and $Y \in \{-1, 1\}$. Assume that X is a continuous random variable with pdf f_X such that $f_X(x) > 0$ for all $x \in \mathbb{R}$. Let $\eta(x) = \Pr(Y = 1 \mid X = x)$ and assume η is a continuous function.

   (a) Let h^* be the Bayes classifier under the 0-1 loss. Show that the conditional Bayes risk under the 0-1 loss is given by

$$R_x(h^*) := \mathbb{E}[\mathbb{1}_{Y \neq h^*(X)} \mid X = x] = \Pr(Y \neq h^*(X) \mid X = x) = \min(\eta(x), 1 - \eta(x)).$$

   Solution: We have already shown in PS2 that the Bayes classifier is

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } \eta(x) \geq (1 - \eta(x)) \\ -1 & \text{otherwise} \end{cases}$$

   We have

$$\begin{aligned}\Pr\left(Y \neq h^*(x) \middle| X = x\right) &= \Pr\left(Y = 1, h^*(X) = -1 \middle| X = x\right) + \Pr\left(Y = -1, h^*(X) = 1 \middle| X = x\right) \\ &= \eta(x)\mathbb{1}_{h^*(x)=-1} + (1 - \eta(x))\mathbb{1}_{h^*(x)=1} \\ &= \min(\eta(x), (1 - \eta(x))).\end{aligned}$$

   (b) For any test point $\widetilde{x} \in \mathbb{R}$, denote $X_{n,(1)} \in \mathbb{R}$ the nearest neighbour of $\widetilde{x}$ in the set $\{X_1, \ldots, X_n\}$, and $Y_{n,(1)}$ its corresponding class. The 1-nearest neighbour (1-NN) classifier is defined as

$$\widehat{h}^{(D_n)}(\widetilde{x}) = Y_{n,(1)}.$$

   Let $(\widetilde{X}, \widetilde{Y})$ be a test sample, with the same distribution as (X, Y), and independent of the training sample D_n. Show that

$$\Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}, X_{n,(1)} = x_{n,(1)}) = \eta(\widetilde{x})(1 - \eta(x_{n,(1)})) + (1 - \eta(\widetilde{x}))\eta(x_{n,(1)}).$$

Solution: We have

$$\begin{aligned}\Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}, X_{n,(1)} = x_{n,(1)}) &= \Pr(\widetilde{Y} = 1, Y_{n,(1)} = -1 \mid \widetilde{X} = \widetilde{x}, X_{n,(1)} = x_{n,(1)}) \\ &\quad + \Pr(\widetilde{Y} = -1, Y_{n,(1)} = 1 \mid \widetilde{X} = \widetilde{x}, X_{n,(1)} = x_{n,(1)}) \\ &= \eta(\widetilde{x})(1 - \eta(x_{n,(1)})) + (1 - \eta(\widetilde{x}))\eta(x_{n,(1)})\end{aligned}$$

using the fact that $(\widetilde{X}, \widetilde{Y})$ is independent of D_n.

(c) Show that, for any $\widetilde{x} \in \mathbb{R}$, $X_{n,(1)}$ converges to $\widetilde{x}$ almost surely as $n \to \infty$.

[Hint: first prove that the convergence holds in probability. Then use the following lemma:
Let $Z_1 \geq Z_2 \geq Z_3 \geq \ldots$ be a monotone decreasing sequence of random variables. If $Z_n \to Z$ in probability, then $Z_n \to Z$ almost surely.]

Solution: We first want to show that, for any $\epsilon > 0$,

$$\Pr(|X_{n,(1)} - \widetilde{x}| > \epsilon) \to 0$$

as n tends to infinity. We have

$$\begin{aligned}\Pr(|X_{n,(1)} - \widetilde{x}| > \epsilon) &= \Pr(|X_i - \widetilde{x}| > \epsilon \text{ for all } i = 1, \ldots, n) \\ &= \Pr(|X - \widetilde{x}| > \epsilon)^n \\ &= \left(1 - \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} f_X(x)dx\right)^n \to 0 \text{ as } n \to \infty\end{aligned}$$

as $1 - \int_{\widetilde{x}-\epsilon}^{\widetilde{x}+\epsilon} f_X(x)dx < 1$. Therefore, $|X_{n,(1)} - \widetilde{x}| \to 0$ in probability. Noting that $|X_{n,(1)} - \widetilde{x}|$ is a monotone decreasing sequence and using the given lemma, we conclude $|X_{n,(1)} - \widetilde{x}| \to 0$ almost surely, or $X_{n,(1)} \to \widetilde{x}$ almost surely.

(d) Give the limit, as $n \to \infty$, of

$$\Pr\left(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}\right).$$

Deduce that

$$R(h^*) \leq \lim_{n\to\infty} \mathbb{E}[R(\widehat{h}^{(D_n)})] \leq 2R(h^*)(1 - R(h^*))$$

where, for a classifier h, $R(h) = \mathbb{E}[\mathbb{1}_{Y \neq h(X)}] = \Pr(Y \neq h(X))$ is the risk under the 0-1 loss.

[Hint: you may use this simplified version of the dominated convergence theorem (see Part A Integration for details).
*Let $(X_1, X_2, \ldots)$ be a **bounded** sequence of random variables such that $X_n \to X$ almost surely. Then $\lim_{n\to\infty} \mathbb{E}[X_n] = \mathbb{E}[X]$.]*

Solution: The first inequality follows by definition of the Bayes classifier. As η is continuous, $X_{n,(1)} \to \widetilde{x}$ implies $\eta(X_{n,(1)}) \to \eta(\widetilde{x})$ almost surely. Therefore, for any fixed $\widetilde{x}$,

$$\begin{aligned} g(\widetilde{x}, X_{n,(1)}) &:= \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}, X_{n,(1)}) \\ &= \eta(\widetilde{x})(1 - \eta(X_{n,(1)})) + (1 - \eta(\widetilde{x}))\eta(X_{n,(1)}) \\ &\to 2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) \end{aligned}$$

almost surely as $n \to \infty$. Using the dominated convergence theorem, for any $\widetilde{x}$

$$\begin{aligned} \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}) &= \mathbb{E}[g(\widetilde{x}, X_{n,(1)})] \\ &\to 2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) \end{aligned}$$

as $n \to \infty$. Note that

$$2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) = 2R_{\widetilde{x}}(h^*)(1 - R_{\widetilde{x}}(h^*))$$

and

$$\mathbb{E}[R(\widehat{h}^{(D_n)})] = \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}))$$

Another application of the dominated convergence theorem yields

$$\begin{aligned} \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X})) &= \mathbb{E}_{\widetilde{X}}[\Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X})] \\ &\to 2\,\mathbb{E}[R_{\widetilde{X}}(h^*)(1 - R_{\widetilde{X}}(h^*))] \end{aligned}$$

as $n \to \infty$. Finally, (using Jensen's inequality, or just developing the expression)

$$\begin{aligned} \mathbb{E}[R_{\widetilde{X}}(h^*)(1 - R_{\widetilde{X}}(h^*))] &\leq \mathbb{E}[R_{\widetilde{X}}(h^*)](1 - \mathbb{E}[R_{\widetilde{X}}(h^*)]) \\ &= R(h^*)(1 - R(h^*)). \end{aligned}$$

François Caron: caron@stats.ox.ac.uk

4. (Surrogate loss functions) Consider a binary classification problem, with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{-1, 1\}$. Let h be a classifier

$$h(x) = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

with discriminant function $f : \mathcal{X} \to \mathbb{R}$ taking values in $\mathcal{F}_f$, the set of functions from $\mathcal{X}$ to $\mathbb{R}$. For a surrogate loss function ψ, let

$$R_\psi(f) = \mathbb{E}[\psi(Yf(X))]$$

be the ψ-risk of the discriminant function f. If $\psi(z) = \mathbb{1}_{z<0}$, then $R_\psi(f) = \Pr(Yf(X) < 0) = \Pr(Y \neq \text{sign}(f(X)))$ is the usual risk under the 0-1 loss. Denote

$$h^*_\psi = \begin{cases} 1 & \text{if } f^*_\psi(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

with

$$f^*_\psi = \arg\min_{f \in \mathcal{F}_f} R_\psi(f)$$

the (optimal) Bayes classifier under the surrogate ψ loss. Denote h^* the Bayes classifier under the 0-1 loss.

(a) For each of the following surrogate functions, find the associated (optimal) discriminant function f^*_ψ and show that $h^*_\psi = h^*$

$$\psi_1(z) = e^{-z}$$
$$\psi_2(z) = (1-z)^2$$
$$\psi_3(z) = \log(1 + e^{-z})/\log(2)$$

Solution: Recall that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

where $\eta(x) = \Pr(Y = 1 \mid X = x)$. It suffices to minimise the conditional ψ-risk, for each x,

$$f^*_\psi(x) = \arg\min_{f(x) \in \mathbb{R}} \mathbb{E}[\psi(Yf(x))|X = x].$$

We have

$$\mathbb{E}[\psi(Yf(x))|X = x] = \psi(f(x))\eta(x) + \psi(-f(x))(1 - \eta(x))$$

Define $g_\psi(u) = \psi(u)\eta(x) + \psi(-u)(1 - \eta(x))$. If ψ is differentiable,

$$g'_\psi(u) = \psi'(u)\eta(x) - \psi'(-u)(1 - \eta(x))$$

This gives

$$\begin{aligned}
g'_{\psi_1}(u) &= -e^{-u}\eta(x) + e^u(1-\eta(x)) = 0 \\
g'_{\psi_2}(u) &= -2(1-u)\eta(x) + 2(1+u)(1-\eta(x)) = 2(1+u) - 4\eta(x) = 0 \\
g'_{\psi_3}(u) &= \frac{1}{\log 2}\left(-\operatorname{sig}(u)\eta(x) + \operatorname{sig}(-u)(1-\eta(x))\right) \\
&= \frac{1}{\log 2}\left(-(1-\operatorname{sig}(u))\eta(x) + \operatorname{sig}(u)(1-\eta(x))\right) \\
&= \frac{1}{\log 2}\left(\operatorname{sig}(u) - \eta(x)\right) = 0
\end{aligned}$$

which gives

$$\begin{aligned}
f^*_{\psi_1}(x) &= \frac{1}{2}\log\frac{\eta(x)}{1-\eta(x)} \\
f^*_{\psi_2}(x) &= 2\eta(x) - 1 \\
f^*_{\psi_3}(x) &= \log\frac{\eta(x)}{1-\eta(x)}
\end{aligned}$$

In all cases, the associated Bayes classifier under the ψ risk is the same as the Bayes classifier under the 0-1 loss. This property of the surrogate loss ψ is sometimes called Bayes consistency.

(b) Let $\mathcal{X} = \mathbb{R}$. Consider the class $\mathcal{H}_f \subset \mathcal{F}_f$ of linear discriminant functions of the form $f(x) = \beta_0 + \beta_1 x$. Denote

$$f^*_{\psi,\mathcal{H}_f} = \operatorname*{arg\,min}_{f\in\mathcal{H}_f} R_\psi(f)$$

the optimal discriminant function **amongst this class** for the surrogate risk ψ, and let $h^*_{\psi,\mathcal{H}_f}$ be the associated classifier. Denote $f^*_{\mathcal{H}_f}$ the optimal discriminant function in this class under the 0-1 loss and $h^*_{\mathcal{H}_f}$ the associated classifier.

Assuming that $\Pr(Y=1) = \Pr(Y=-1) = 1/2$, Show that, for the surrogate loss ψ_2, we have $f^*_{\psi_2,\mathcal{H}_f}(\mathbb{E}(X)) = 0$. Deduce that in general, $h^*_{\psi_2,\mathcal{H}_f} \neq h^*_{\mathcal{H}_f}$.

Solution: Note that for a discriminant function of the form $f(x) = \beta_0 + \beta_1 x$, the decision boundary is defined by $x = -\beta_0/\beta_1$. Let

$$J(\beta_0,\beta_1) = \mathbb{E}[(1 - Y(\beta_0+\beta_1 X)^2)] = \mathbb{E}[(Y - (\beta_0+\beta_1 X))^2)]$$

Differentiating wrt β_0, we obtain

$$\frac{\partial J}{\partial \beta_0} = -2\,\mathbb{E}[(Y - (\beta_0+\beta_1 X))] = 2\beta_0 + 2\beta_1\,\mathbb{E}[X]$$

Setting this to 0 gives

$$-\frac{\beta_0}{\beta_1} = \mathbb{E}[X]$$

Hence $x = -\frac{\beta_0}{\beta_1} = \mathbb{E}[X]$ is the decision boundary between the two classes under the ψ_2 surrogate loss. This is not the case in general under the 0-1 loss. For example, taking as distribution for class 1 $g_1(x) = \mathbb{1}_{x\in(0,1)}$ and for class -1 $g_{-1}(x) = a^{-1}\mathbb{1}_{x\in(-a,0)}$, $a > 1$, then the decision boundary is $x = 0$ under the 0-1 loss, but is $\mathbb{E}[X] = (1-a)/4 \neq 0$ under the ψ_2 surrogate loss. (this is analogous to the example shown in the lectures with the outlier).

5. (Coding: Classification trees and Random Forests)

   The code below loads the wine dataset, that we have already analysed with Linear Discriminant Analysis in Problem Sheet 2.

   (a) We first consider a classification tree.

   (i) Fit a full classification tree to the data and print the tree. Relate the classification rule discovered there to the parameters estimated by LDA.

   (ii) For trees with different number of leaf nodes, report the risk, approximated using 5-fold cross-validation (use the function `cross_val_score`).

   (iii) For the number of leaf nodes that minimises the cross-validation risk, refit the tree on the whole dataset, and plot the tree.

   (b) We now consider a random forest classifier.

   (i) For 500 trees, and different number of features $m = 1, \ldots, 6$ considered for each tree (option max_features in the function `RandomForestClassifier`), calculate and plot the out-of-bag misclassification error.

   (ii) For 500 trees and $m = 3$ features used per tree, plot the variable importances.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler # to normalise the data
from sklearn.datasets import load_wine
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
plt.rcParams['figure.dpi'] = 200

# Load wine data
(X,y) = load_wine(return_X_y = True, as_frame = True)
Xy = pd.concat([X, y], axis=1)

(n, p) = X.shape # number of examples and dimension of each example
K = y.unique().size # number of classes
print('n=', n, ', p=', p, ', K=', K)
X.head()
```

Solution: See Python notebook. For a solution in R, see the solution by Oliver Cobb: http://www.stats.ox.ac.uk/~palamara/teaching/SML19/PS4_Q4Q5_sol.pdf

6. (Coding: Neural Networks)

   In this question, we will investigate the use of shallow and multi-layer neural networks for multiclass classification. The code below loads the digits dataset, that we considered in Problem Sheet 2. Recall that the objective is to classify 64×64 images of handwritten digits from 0 to 9. The provided code splits the dataset into a training and test set, and standardises the data.

   (a) Fit a logistic regression model on the training data, and report the accuracy on the training and test sets.

   (b) We now consider a shallow neural network (one hidden layer) with $m = 50$ neurons and tanh activation function. For different values $n_b = 1, 32, 100, 898$ of the batch size (parameter `batch_size` in the function `MLPClassifier`), report the value of the objective function as a function of the number of epochs (where one epoch corresponds to a pass through the entire dataset), when using a stochastic gradient algorithm (you should set `solver='SGD'` in the `MLPClassifier` function).

   (c) Consider now changing the structure of the shallow neural network, that is the number m of neurons. For $m = 5, 10, 20, 100, 200, 300$, fit the neural network parameters. Report the accuracy on the test and training sets as a function of the number of neurons m.

   (d) Consider now a feedforward neural network, with the following number of neurons at the different hidden layers: (100,100,10). Fit the parameters of this model, and report the accuracy on the test and training data.

   (e) (Optional) Try to change the structure of the network, the activation function, or the parameters of the optimiser (e.g. `max_iter, tol, n_iter_no_change`) to improve the accuracy.

```
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler # to normalise the data

# Load data
(X, y) = load_digits(return_X_y=True, as_frame=True, n_class = 10)
```


François Caron: caron@stats.ox.ac.uk

```
# Plot some images in the dataset
for i in range(30):
    plt.subplot(3, 10, i + 1)
    image = X.loc[i,:].to_numpy()
    plt.imshow(image.reshape(8, 8), cmap=plt.cm.gray)
    plt.title(y[i])
    plt.xticks(())
    plt.yticks(())

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
print('Training:', X_train.shape)
print('Test:', X_test.shape)

# Scale
scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)
```

Solution: See Python notebook

7. (Optional. Coding: Convolutional Neural Networks) The attached Python notebook implements a convolutional neural network for classifying 28-by-28 images of handwritten digits from the MNIST dataset, a popular machine learning benchmark for multiclass classification. It uses TensorFlow, one of a number of popular deep learning packages.

   (a) If needed, install Tensorflow; run the code.

   (b) Change the parameters of the convolutional neural networks (e.g. number, size of the filter, number of convolutional layers, etc.), and report the accuracy on the training/test data.