2EL1730: Machine Learning

CentraleSupélec - 2A

Individual Graded Assignment

27/09/2024

CentraleSupélec

Edward Lucyszyn

Contents

Exercise I: True/ False

1. **False.** Principal Component Analysis (PCA) is a linear dimensionality reduction method because the result $\mathbf{X_{PCA}}$ can be expressed as a linear expression depending on the data $\mathbf{X}$, a matrix product of this form:
$$\mathbf{X_{PCA}} = \mathbf{AX} + \mathbf{B}.$$

2. **True.** Clustering is a subjective task because it can lead to different clusters depending on the algorithm used. For example, in a dataset filled with 2 circular lines of nodes, k-means and Spectral Clustering give different results. Moreover, the number of clusters k is subjective.

3. **True.** The Kernel trick with SVM is to avoid the direct computation of $\phi(x_i)$ by computing directly the inner product $\langle \phi(x_1), \phi(x_2) \rangle = k(x_1, x_2)$. So, there is no need to have the expression of ϕ. Also, the output of a Kernel function must be in a Hilbert space; the function does not have to be of finite dimension.

4. **False.** The main purpose of Fisher Discriminant Analysis (FDA) is to reduce dimensionality while enforcing class separation. One of its applications is to perform classification, but it is not its main purpose.

5. **False.** For example, sometimes the phenomenon of overfitting can occur. In this case, even if the number of entry parameters increases, the model will not perform well on unseen data.

6. **True.** Instead of computing the derivative for each parameter of the data sample, Stochastic Gradient Descent allows for calculating the gradient on a single point at each step. For an input with, for example, 1,000,000 points, the number of calculations at each step can be reduced by 1,000,000.

7. **True.** A 1-NN classifier has higher variance than a 3-NN classifier because the 1-NN is sensitive to mislabeled data.

8. **False.** If A is a rectangular matrix of shape $(n, q) \in \mathbb{N}^2$ with $n \neq q$, then $A^\intercal A$ is of size (q, q) and $AA^\intercal$ is of size (n, n). Thus, since the eigenvectors are not of the same dimension, they cannot be the same.

9. **False.** In AdaBoost, the error associated with each hypothesis is computed as the weighted sum of misclassified examples, not as a simple ratio, because each example has its own weight that is updated after each step.

10. **True.** For example, if X represents the data and θ the set of parameters, then the posterior distribution to maximize is $f(\theta|X)$. Then, with Bayes' formula, $f(\theta|X) \propto f(X|\theta)\pi(\theta)$ with $\pi(\theta)$ being the prior distribution and $f(X|\theta)$ the likelihood. Thus, it shows that it considers the parameters of the model as random variables following a specific prior distribution.

Exercise II: Learning, Model Selection and Evaluation

By direct calculations:

$$\begin{aligned}\text{MSE}(\hat{f}(x)) &= \mathbb{E}[(y - \hat{f}(x))^2] \\ &= \mathbb{E}[y^2 + \hat{f}^2(x) - 2y\hat{f}(x)] \\ &= \mathbb{E}[f^2(x) + \epsilon^2 + 2\epsilon f(x) + \hat{f}^2(x) - 2f(x)\hat{f}(x) - 2\epsilon\hat{f}(x)].\end{aligned}$$

Since ϵ is a zero-mean noise and has a variance of σ^2, $\mathbb{E}(\epsilon) = 0$ and $\text{Var}(\epsilon) = \mathbb{E}(\epsilon^2) = \sigma^2$. Furthermore, $f(x)$ is constant. Thus,

$$\begin{aligned}\text{MSE}(\hat{f}(x)) &= \mathbb{E}[(\hat{f}(x) - f(x))^2] + \sigma^2 \\ &= \text{Var}(\hat{f}(x) - f(x)) + (\text{Bias}(\hat{f}(x)))^2 + \sigma^2 \\ &= \text{Var}(\hat{f}(x)) + (\text{Bias}(\hat{f}(x)))^2 + \sigma^2.\end{aligned}$$

So,

$$\boxed{\text{MSE}(\hat{f}(x)) = \text{Var}(\hat{f}(x)) + (\text{Bias}(\hat{f}(x)))^2 + \sigma^2.}$$

Exercise III: Linear and Logistic Regression

(a) The cost function is:

$$J(\mathbf{w}) = ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + ||\Gamma\mathbf{w}||_2^2.$$

We recall the formula for a square matrix $\mathbf{A}$ and a vector $\mathbf{w}$:

$$\frac{\partial}{\partial \mathbf{w}}(\mathbf{w} \mapsto ||\mathbf{A}\mathbf{w}||_2^2)(\mathbf{w}) = \frac{\partial}{\partial \mathbf{w}}(\mathbf{w} \mapsto \mathrm{Tr}(\mathbf{x}^\intercal \mathbf{A}^\intercal \mathbf{A}\mathbf{w}))(\mathbf{w}) = 2\mathbf{A}^\intercal\mathbf{A}\mathbf{w};$$

and,

$$\frac{\partial}{\partial \mathbf{w}}(\mathbf{w} \mapsto \mathrm{Tr}(\mathbf{A}\mathbf{w}))(\mathbf{w}) = \mathbf{A}^\intercal\mathbf{w}.$$

Furthermore,

$$\begin{aligned} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 &= \mathrm{Tr}((\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal(\mathbf{X}\mathbf{w} - \mathbf{y})) \\ &= \mathrm{Tr}(\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w} - \mathbf{y}^\intercal\mathbf{X}\mathbf{w} - \mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y} + \mathbf{y}^\intercal\mathbf{y}) \\ &= ||\mathbf{X}\mathbf{w}||_2^2 - 2\,\mathrm{Tr}(\mathbf{y}^\intercal\mathbf{X}\mathbf{w}) + ||\mathbf{y}||_2^2). \end{aligned}$$

Thus,

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}) = 2\mathbf{X}^\intercal\mathbf{X}\mathbf{w} - 2\mathbf{X}^\intercal\mathbf{y} + 2\boldsymbol{\Gamma}^\intercal\boldsymbol{\Gamma}\mathbf{w}.$$

The minimum appears when the derivative is cancelled, so we have:

$$\frac{\partial}{\partial \mathbf{w}^*} J(\mathbf{w}^*) = 0 \iff \boxed{(\mathbf{X}^\intercal\mathbf{X} + \boldsymbol{\Gamma}^\intercal\boldsymbol{\Gamma})\mathbf{w}^* = \mathbf{X}^\intercal\mathbf{y}.}$$

(b) With (a), a sufficient and necessary condition on Γ guaranteeing that $J(\mathbf{w})$ has a unique minimum $\mathbf{w}^*$ is:

$$\boxed{\mathbf{X}^\intercal\mathbf{X} + \boldsymbol{\Gamma}^\intercal\boldsymbol{\Gamma} \in \mathcal{GL}_d(\mathbb{R}).}$$

In this case, we have:

$$\boxed{\mathbf{w}^* = (\mathbf{X}^\intercal\mathbf{X} + \boldsymbol{\Gamma}^\intercal\boldsymbol{\Gamma})^{-1}\mathbf{X}^\intercal\mathbf{y}.}$$

(c) Using Bayes formula:

$$f(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto f(\mathbf{w}) f(\mathbf{y}|\mathbf{x}, \mathbf{w}),$$

where $f(\mathbf{w})$ is the normal distribution $\mathcal{N}(0, \Sigma)$. Consequently:

$$f(\mathbf{w}) \propto \exp(-\frac{1}{2}\mathbf{w}^\top\Sigma^{-1}\mathbf{w}).$$

Then, since Σ is a positive-definite matrix, $\exists \mathbf{S}$ such that $\Sigma = \mathbf{S}\mathbf{S}^\top$. Assuming a linear regression model (logic assumption because this is a famous model and it will lead to the ridge regression problem), we have:

$$f(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})}{2}\right) \propto \exp(-\frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2).$$

Using the log, maximizing the posterior regression leads to maximizing the quantity:

$$\boxed{J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + ||\mathbf{S}^{-1}\mathbf{w}||_2^2.}$$

Exercise IV: Dimension Reduction

(a) Let J be the cost function defined as:

$$J : \mathbf{w} \mapsto \frac{\mathbf{w}^\top \mathbf{S_b} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}.$$

Then, to find the maximum of J, we take the derivative and set it to zero.

$$\frac{d}{d\mathbf{w}}[J(\mathbf{w})] = \frac{d}{d\mathbf{w}}\left[\frac{\mathbf{w}^\top \mathbf{S_b} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}\right] = 0$$

$$\iff \left(\left[\mathbf{w}^\top \mathbf{S_w} \mathbf{w}\right] \frac{d\left[\mathbf{w}^\top \mathbf{S_b} \mathbf{w}\right]}{d\mathbf{w}} - \left[\mathbf{w}^\top \mathbf{S_b} \mathbf{w}\right] \frac{d\left[\mathbf{w}^\top \mathbf{S_w} \mathbf{w}\right]}{d\mathbf{w}}\right) / \left(\mathbf{w}^\top \mathbf{S_w} \mathbf{w}\right)^2 = 0$$

$$\iff \left(\left[\mathbf{w}^\top \mathbf{S_w} \mathbf{w}\right] 2\mathbf{S_b} \mathbf{w} - \left[\mathbf{w}^\top \mathbf{S_b} \mathbf{w}\right] 2\mathbf{S_w} \mathbf{w}\right) / \left(\mathbf{w}^\top \mathbf{S_w} \mathbf{w}\right)^2 = 0$$

$$\iff \left[\frac{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}\right] \mathbf{S_b} \mathbf{w} - \left[\frac{\mathbf{w}^\top \mathbf{S_b} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}\right] \mathbf{S_w} \mathbf{w} = 0$$

$$\iff \mathbf{S_b} \mathbf{w} - J(\mathbf{w}) \mathbf{S_w} \mathbf{w} = 0$$

By setting $\lambda = \left[\frac{\mathbf{w}^\top \mathbf{S_b} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}\right]$, we obtain the *Generalized eigenvalue problem*:

$$\boxed{{\mathbf{S_w}}^{-1} \mathbf{S_b} \mathbf{w} = \lambda \mathbf{w}.}$$

(b) In this problem, $\mathbf{w}$ is one of the eigenvector that correspond to the eigenvalue λ. Since, $\lambda = \left[\frac{\mathbf{w}^\top \mathbf{S_b} \mathbf{w}}{\mathbf{w}^\top \mathbf{S_w} \mathbf{w}}\right] = J(\mathbf{w})$, the solution to the problem will be one of the eigenvector corresponding to the highest eigenvalue of the *Generalized eigenvalue problem*.

(c) The first step of the Pirncipal Component Analysis (PCA) algorithm is to center and standardize the matrix. By setting,

$$\mathbf{X_1} = \begin{bmatrix} 0.4669 \\ 0.2097 \\ 0.6252 \\ 0.1832 \\ 1.0298 \end{bmatrix}; \quad \mathbf{X_2} = \begin{bmatrix} 0.9492 \\ 0.3071 \\ 0.1352 \\ 0.5152 \\ 0.2614 \end{bmatrix};$$

we have, $\mu(X_1) = 0.00716, \mu(X_2) = 0.43362, \sigma(X_1) = 0.59103, \sigma(X_2) = 0.28537$. Then, computing the covariance matrix we the standardized matrix $\mathbf{X}$:

$$\boxed{\mathbf{V} = \begin{bmatrix} 1.25 & 0.40461055 \\ 0.40461055 & 1.25 \end{bmatrix}.}$$

(d) The eigenvalues are:

$$\boxed{\sigma(\mathbf{V}) = \{1.65461055, 0.84538945\}.}$$

(e) The variance of the principal component accounts for $\boxed{66\%}$. It is not enough. We try to reach 80% of 90%.

Exercise V: Nearest Neighbours and kd-Trees

(a) Here is the code.

```python
#%%
## Importing libraries

import numpy as np

#%%
## Exercice V: Nearest Neighbours and kd-Trees

# Given coordinates
coordinates = [
    0.28, 0.13, 0.65, 0.31, 0.61, 0.56, 0.75, 0.15,
    0.71, 0.12, 0.97, 0.28, 0.38, 0.97, 0.97, 0.15,
    0.03, 0.13, 0.96, 0.44, 1.00, 0.97, 0.43, 0.79,
    0.47, 0.00, 0.04, 0.81, 0.10, 0.44, 0.42, 0.26,
    0.93, 0.31, 0.33, 0.57, 0.44, 0.18, 0.07, 0.44,
    0.26, 0.50, 0.64, 0.49, 0.42, 0.29, 0.06, 0.78,
    0.55, 0.65, 0.40, 0.75, 0.55, 0.33, 0.77, 0.93,
    0.90, 0.65, 0.40, 0.94, 0.29, 0.35, 0.88, 0.12,
    0.48, 0.34, 0.95, 0.47, 0.85, 0.05, 0.56, 0.15,
    0.86, 0.83, 0.36, 0.74, 0.09, 0.69, 0.47, 0.34,
    0.50, 0.98, 0.98, 0.69, 0.67, 0.05, 0.23, 0.54,
    0.37, 0.78, 0.67, 0.35, 0.52, 0.36, 0.03, 0.54,
    0.72, 0.38, 0.72, 0.22, 0.11, 0.62, 0.65, 0.22,
    0.41, 0.09, 0.23, 0.35, 0.38, 0.65, 0.33, 0.73,
    0.95, 0.76, 0.83, 0.52, 0.44, 0.32, 0.15, 0.01,
    0.31, 0.65, 0.78, 0.33, 0.85, 0.23, 0.83, 0.21,
    0.60, 0.82, 0.33, 0.75, 0.05, 0.28, 0.70, 0.32,
    0.65, 0.17, 0.56, 0.65, 0.96, 0.47, 0.55, 0.14,
    0.28, 0.68, 1.00, 0.11, 0.53, 0.84, 0.50, 0.75,
    0.88, 0.36, 0.88, 0.97, 0.72, 0.90, 0.62, 0.31,
    0.64, 0.53, 0.64, 0.33, 0.79, 0.37, 0.85, 0.14,
    0.63, 0.80, 0.95, 0.14, 0.50, 0.25, 0.06, 0.64,
    0.82, 0.75, 0.42, 0.21, 0.42, 0.98, 0.65, 0.78,
    0.87, 0.16, 0.97, 0.80, 0.63, 0.46, 0.45, 0.04,
    0.01, 0.98, 0.99, 0.14, 0.53, 0.54, 0.57, 0.47
]

# Reshape the coordinates into a numpy matrix
data = np.array(coordinates).reshape(-1, 4)
```

```
    #%%

    def sorted_index(list):
        """Return the index of the median sorted value of a list."""
        return sorted(range(len(list)), key=lambda i: list[i])

    def choose_pivot_index(list):
        """Return the index of the pivot of a list."""
        return len(list) // 2

    def split_data(points, points_index, depth):
        """Split the data into two parts according to the pivot of the \
+       ↪ points."""
        # Get the dimension of the points
        dimension = points.shape[1]

        # Sort the points according to the median
        sorted_points = \
+       ↪ list(np.array(points_index)[sorted_index(points[points_index][:, \
+       ↪ depth % dimension])])

        # Get the index of the pivot
        pivot_index = choose_pivot_index(sorted_points)

        # Get the median point
        pivot_point = sorted_points[pivot_index]

        # Split the points into two parts
        left_points = sorted_points[:pivot_index]
        right_points = sorted_points[pivot_index + 1:]

        return pivot_point, left_points, right_points

    def create_kd_tree(points, points_index=-1, depth=0):
        """Create a kd-tree from a set of points."""
        if points_index == -1:
            points_index = list(range(len(points)))

        # If there are no points, return None
        if len(points_index) == 0:
            return None
```

```
        # Split the points into two parts
        pivot_point, left_points, right_points = split_data(points, \
+       ↪ points_index, depth)

        # Create the node of the tree
        tree = {
            'point': pivot_point,
            'left': create_kd_tree(points, left_points, depth + 1),
            'right': create_kd_tree(points, right_points, depth + 1)
        }

        return tree

    #%%
    # Order of the tree

    def compute_tree_order(tree):
        """Compute the order of a binary tree with dictionaries."""
        if tree['left'] == None and tree['right'] == None:
            return [tree['point']]
        elif tree['left'] == None and tree['right'] != None:
            return [tree['point']] + compute_tree_order(tree['right'])
        elif tree['left'] != None and tree['right'] == None:
            return [tree['point']] + compute_tree_order(tree['left'])
        else:
            return [tree['point']] + compute_tree_order(tree['left']) + \
+           ↪ compute_tree_order(tree['right'])

    result = list(np.array(compute_tree_order(create_kd_tree(data))) + 1)

    print(result)
```

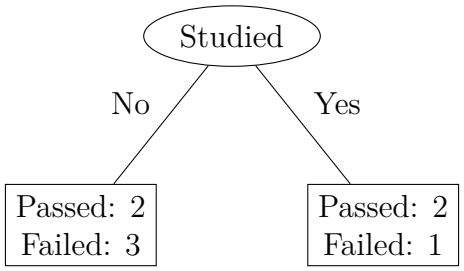
The output produced by this algorithm, utilizing the coordinates from the PDF file, is as follows (nodes are indexed from 1 to 50): [50, 8, 27, 44, 10, 30, 24, 7, 12, 34, 16, 1, 17, 5, 23, 28, 26, 20, 11, 38, 46, 4, 37, 49, 21, 31, 36, 14, 22, 25, 48, 18, 9, 35, 3, 32, 42, 39, 47, 6, 33, 19, 45, 15, 13, 40, 43, 2, 29, 41].

(b) At each step of the decision tree, the algorithm selects the optimal split, which involves examining every possible splitting value for each of the d features. Consequently, considering the n data points and assuming linear time for sorting real numbers, the complexity of this step is $O(nd)$. This process must be repeated for every layer of the tree, occurring h times. Therefore, the final complexity is indeed $O(ndh)$.

Exercise VI: Tree-based Methods and Ensembles

(a) We opt to split the decision tree to maximize information gain, i.e., minimize entropy.

For entropy calculations, according to [2], we have:

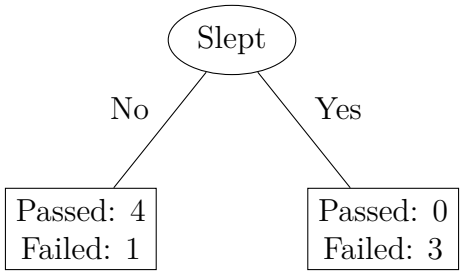


$$I(\texttt{Studied = Yes}) = -\frac{2}{3}\log_2(\frac{2}{3}) - \frac{1}{3}\log_2(\frac{1}{3}) = 0.918;$$

$$I(\texttt{Studied = No}) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0.971;$$

$$\boxed{I(\texttt{Studied}) = \frac{3}{8}I(\texttt{Studied = Yes}) + \frac{5}{8}I(\texttt{Studied = No}) = 0.951.}$$

For the entropy of "Slept":

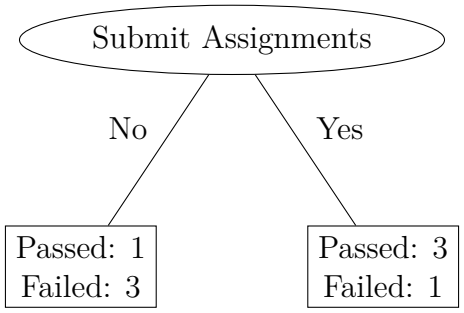


$$I(\texttt{Slept = Yes}) = -\frac{0}{3}\log_2(\frac{0}{3}) - \frac{3}{3}\log_2(\frac{3}{3}) = 0;$$

$$I(\texttt{Slept = No}) = -\frac{1}{5}\log_2(\frac{1}{5}) - \frac{4}{5}\log_2(\frac{4}{5}) = 0.722;$$

$$\boxed{I(\texttt{Slept}) = \frac{3}{8}I(\texttt{Slept = Yes}) + \frac{5}{8}I(\texttt{Slept = No}) = 0.451.}$$

For the entropy of "Submit Assignments":



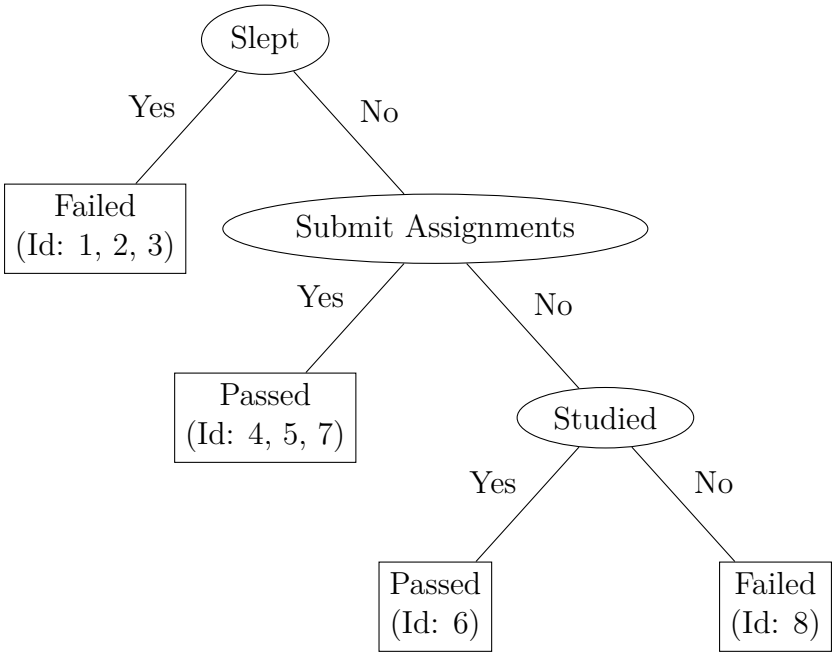
$$I(\texttt{Submit Assignments = Yes}) = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) = 0.811;$$
$$I(\texttt{Submit Assignments = No}) = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) = 0.811;$$

$$I(\texttt{Submit Assignments}) = \frac{4}{8}I(\texttt{Submit Assignments = Yes}) + \frac{4}{8}I(\texttt{Submit Assignments = No}) = 0.811.$$

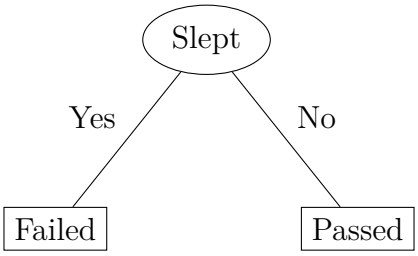
Thus, the feature to split on that minimizes the entropy is "Slept". The information gain is the following:

$$\Delta_{\text{info}}(\texttt{Slept}) = I(\texttt{Parent}) - I(\texttt{Slept}) = 1 - 0,451 = 0,549.$$

(b) At each node we choose to split on the feature that maximizes the information gain. We obtain the following decision tree.



(c) If we construct a one-split tree with the split identified in part (a), we have the following decision tree.



In the first iteration, AdaBoost assigns a weight of $D_1(i) = \frac{1}{8}$ to each student $i \in [\![1, 8]\!]$. With this first classifier, there is only one error. Thus the error is:

$$\epsilon_1 = \frac{1}{8}.$$

Then, according to the algorithm described in [1] we compute the gradient step:

$$\alpha_1 = \frac{1}{2}\ln(\frac{1-\epsilon_1}{\epsilon_1}) = \frac{1}{2}\ln(7).$$

We then update the weights of the examples:

$$\forall i \in [\![1, 8]\!], \quad D_2(i) = \frac{D_1(i)e^{-\alpha_1 y_i h_1(x_i)}}{Z_1}$$

where Z_1 is the normalization factor equal to $2\sqrt{\epsilon_1(1-\epsilon_1)} = \frac{1}{4}\sqrt{7}$.
At the end,

$$\boxed{\forall i \in [\![1, 7]\!], \quad D_2(i) = \frac{1}{14}; \quad D_2(8) = \frac{1}{2}.}$$

Exercise VII: Support Vector Machines

• We write the corresponding Kernel function associated to Φ.

$\forall \mathbf{x} = (x_1, x_2) \in \mathbb{R}^2, \forall \mathbf{y} = (y_1, y_2) \in \mathbb{R}^2$:

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= \langle \Phi(x_1, x_2), \Phi(y_1, y_2) \rangle \\ &= 2x_1y_1 + 2x_1x_2y_1y_2 + 1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 \\ &= (x_1y_1 + x_2y_2 + 1)^2 \\ &\boxed{= (\mathbf{x}^T\mathbf{y} + 1)^2.} \end{aligned}$$

This a polynomial Kernel of order 2 that is convex and differentiable.

• Let's try to give the expression of the corresponding SVM solution. We define:

$$\mathbf{x}^1 = (1,1); \quad \mathbf{x}^2 = (-1,_1); \quad \mathbf{x}^3 = (1,-1); \quad \mathbf{x}^4 = (-1,1); \quad y^1 = 1; \quad y^2 = 1; \quad y^3 = -1;$$

$$y^4 = -1; \quad \mathbf{w} = \sum_{i=1}^{4} \alpha_i y^i \Phi(\mathbf{x}^i) \text{ with } (\alpha_i)_{1 \leq i \leq 4} \in \mathbb{R}^4.$$

We now want to find the optimal $(\alpha_i^*)_{1 \leq i \leq 4}$ that minimize the following function:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^i y^j K\left(x^i, x^j\right),$$

with the condition,

$$\sum_{i=1}^{n} \alpha_i y^i = \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0.$$

With observe that:

$$\forall i \in [\![1,4]\!], \quad K(\mathbf{x^i}, \mathbf{x^i}) = 9; \quad \forall (i,j) \in [\![1,4]\!]^2, \text{ s.t. } i \neq j, \quad K(\mathbf{x^i}, \mathbf{x^i}) = 1.$$

By direct calculus,

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_1\alpha_2 - \alpha_3\alpha_4 + \alpha_1\alpha_3 + \alpha_1\alpha_4 + \alpha_2\alpha_3 + \alpha_2\alpha_4 - \frac{1}{2}\alpha_1^2 - \frac{1}{2}\alpha_2^2 - \frac{1}{2}\alpha_3^2 - \frac{1}{2}\alpha_4^2.$$

By using the condition on alphas:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \alpha_1}(\mathbf{w}, b, \alpha) &= 1 - \alpha_2 + \alpha_3 + \alpha_4 - 9\alpha_1 = 1 - 8\alpha_1; \\ \frac{\partial \mathcal{L}}{\partial \alpha_2}(\mathbf{w}, b, \alpha) &= 1 - \alpha_1 + \alpha_3 + \alpha_4 - 9\alpha_2 = 1 - 8\alpha_2; \\ \frac{\partial \mathcal{L}}{\partial \alpha_3}(\mathbf{w}, b, \alpha) &= 1 - \alpha_4 + \alpha_1 + \alpha_2 - 9\alpha_3 = 1 - 8\alpha_3; \\ \frac{\partial \mathcal{L}}{\partial \alpha_4}(\mathbf{w}, b, \alpha) &= 1 - \alpha_3 + \alpha_1 + \alpha_2 - 9\alpha_4 = 1 - 8\alpha_4. \end{aligned}$$

This means that, by cancelling the partial derivatives,

$$\boxed{\forall i \in [\![1,4]\!], \quad \alpha_i^* = \frac{1}{8}.}$$

Since K is a convex differentiable function, any point where the gradient is 0 is a minimum; so this indeed the minimum of $\mathcal{L}$.

After,

$$\boxed{\mathbf{w}^* = \frac{1}{8}(\Phi(\mathbf{x^1}) + \Phi(\mathbf{x^2}) - \Phi(\mathbf{x^3}) - \Phi(\mathbf{x^4})) = \frac{1}{2}\begin{pmatrix} 0 \\ \sqrt{2} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.}$$

• Then,

$$\boxed{b^* = -\frac{\max_{i:y^i=-1} \langle \mathbf{w}^*, \mathbf{x}^i \rangle + \min_{i:y^i=+1} \langle \mathbf{w}^*, \mathbf{x}^i \rangle}{2} = 0.}$$

Thus, the equation of the separating hyperplane for $\mathbf{x} \in \mathbb{R}^6$ is:

$$\mathbf{w}^T\mathbf{x} + b^* = 0 \iff \boxed{x_2 = 0.}$$

The two hyperplane through the support vectors are given by the equation $\mathbf{w}^T\mathbf{x} + b^* = \pm 1$, which give:

$$\boxed{x_2 = \sqrt{2} \text{ and } x_2 = -\sqrt{2}.}$$

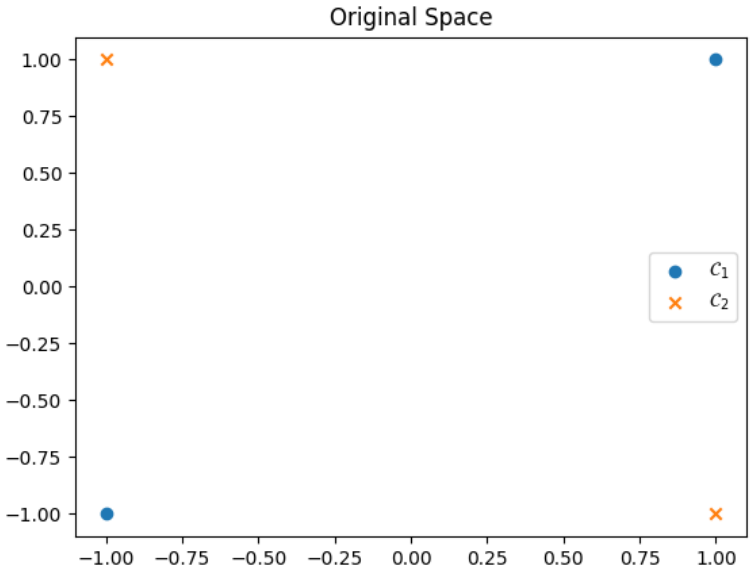


Figure 1: The plot on the original space.

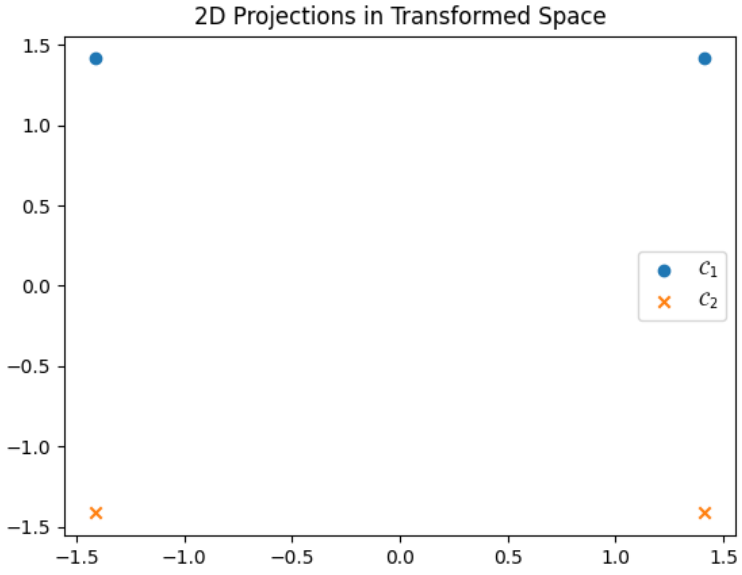


Figure 2: The plot of the 2D projections onto the transformed space.

Exercise VIII: Unsupervised Learning and Clustering

The main distinction between the k-means and k-means++ algorithms lies in the initialization step. While the k-means algorithm randomly selects the positions of centroids, the k-means++ algorithm employs a more intelligent initialization strategy. The first centroid is chosen randomly from the set of points. Subsequently, each succeeding centroid c_{i+1} is chosen from the remaining points. The point x_j is selected with the probability:

$$\mathbb{P}(x_j = c_{i+1}) = \frac{d(x_j, c_i)^2}{\sum_{k,\text{remaining points}} d(x_k, c_i)^2},$$

where $d(x_j, c_i)$ represents the distance between centroid c_i and point x_j. This initialization strategy ensures that points likely to be sufficiently distant from each other.

The quality of the solution depends on the initialization. Choosing centroids randomly may lead the algorithm to converge to a good local minimum rather than the global minimum. Additionally, it can reduce the number of iterations required for convergence. This can be particularly useful in cases involving a large number of points, where each step takes a considerable amount of time to compute, and where local minima are more likely to exist.

Exercise IX: Probabilistic Classifiers: LDA and QDA

(a) $\forall k \in [\![1, K]\!]$,

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu_k)^\intercal \Sigma^{-1}(\mathbf{x}-\mu_k)).$$

Since $t \mapsto \log(t)$ is an increasing function:

$$\begin{aligned}
\hat{C}(\mathbf{x}) &= \underset{k}{\operatorname{argmax}}\, f_k(\mathbf{x})\pi_k \\
&= \underset{k}{\operatorname{argmax}}\, \log f_k(\mathbf{x})\pi_k \\
&= \underset{k}{\operatorname{argmax}}\, \big[-\frac{1}{2}\mathbf{x}^\intercal\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal\Sigma^{-1}\mu_k + \frac{1}{2}\mu_k^\intercal\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_k^\intercal\Sigma^{-1}\mu_k + \log(\pi_k) + \underbrace{\text{Cst}}_{\text{does not depend on } k} \big] \\
&= \underset{k}{\operatorname{argmax}}\, \big[\mathbf{x}^\intercal\Sigma^{-1}\mu_k + \mu_k^\intercal\Sigma^{-1}\mathbf{x} - \mu_k^\intercal\Sigma^{-1}\mu_k + 2\log(\pi_k)\big].
\end{aligned}$$

Using the fact that $\mu_k^\intercal\Sigma^{-1}\mathbf{x}$ is a scalar and Σ is symmetric (because it is a covariance matrix):

$$\mathbf{x}^\intercal\Sigma^{-1}\mu_k = \mu_k^\intercal(\Sigma^{-1})^\intercal\mathbf{x} = \mu_k^\intercal\Sigma^{-1}\mathbf{x}.$$

Thus,

$$\boxed{\hat{C}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\, \big[2\mu_k^\intercal\Sigma^{-1}\mathbf{x} - \mu_k^\intercal\Sigma^{-1}\mu_k + 2\log(\pi_k)\big].}$$

(b) In the case $K = 2$, the boundary equation is when $f_1(\mathbf{x}) = f_2(\mathbf{x})$, which gives:

$$\begin{aligned}
f_1(\mathbf{x}) = f_2(\mathbf{x}) &\iff 2\mu_1^\intercal\Sigma^{-1}\mathbf{x} - \mu_1^\intercal\Sigma^{-1}\mu_1 + 2\log(\pi_1) = 2\mu_2^\intercal\Sigma^{-1}\mathbf{x} - \mu_2^\intercal\Sigma^{-1}\mu_2 + 2\log(\pi_2) \\
&\iff \boxed{(\mu_1^\intercal - \mu_2^\intercal)\Sigma^{-1}\mathbf{x} = \frac{1}{2}(\mu_1^\intercal\Sigma^{-1}\mu_1 - \mu_2^\intercal\Sigma^{-1}\mu_2) + \log(\frac{\pi_2}{\pi_1}).}
\end{aligned}$$

(c) The MAP rule becomes:

$$\begin{aligned}
\hat{C}(\mathbf{x}) &= \underset{k}{\operatorname{argmax}}\, f_k(\mathbf{x})\pi_k \\
&= \underset{k}{\operatorname{argmax}}\, \log f_k(\mathbf{x})\pi_k \\
&= \underset{k}{\operatorname{argmax}}\, \big[-\frac{1}{2}\mathbf{x}^\intercal\Sigma_k^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal\Sigma_k^{-1}\mu_k + \frac{1}{2}\mu_k^\intercal\Sigma_k^{-1}\mathbf{x} - \frac{1}{2}\mu_k^\intercal\Sigma_k^{-1}\mu_k - \log(\pi_k) - \frac{1}{2}\log|\Sigma_k| \big] \\
&= \boxed{\underset{k}{\operatorname{argmax}}\, \big[-\frac{1}{2}\mathbf{x}^\intercal\Sigma_k^{-1}\mathbf{x} + \mu_k^\intercal\Sigma_k^{-1}\mathbf{x} - \frac{1}{2}\mu_k^\intercal\Sigma_k^{-1}\mu_k - \log(\pi_k) - \frac{1}{2}\log|\Sigma_k| \big].}
\end{aligned}$$

(d) QDA stands for "Quadratic Discriminant Analysis." The term "Quadratic" refers to the presence of the term

$$-\frac{1}{2}\mathbf{x}^\intercal\Sigma_k^{-1}\mathbf{x}$$

in the MAP rule. This term is canceled out when $\forall k, \Sigma_k = \Sigma$, but in the case of non-equal covariance assumption, it cannot be eliminated. Therefore, for example, in the scenario of non-equal covariance with $K = 2$, the decision boundary equation becomes a quadratic equation, not a linear equation, as seen in question (b).

Bibliography

[1] *AdaBoost*. Wikipedia, 2024. URL: https://en.wikipedia.org/wiki/AdaBoost.

[2] Pang-Ning Tan et al. *Introduction to Data Mining (Second Edition)*. University of Minnesota, 2021. URL: https://www-users.cse.umn.edu/~kumar001/dmbook/index.php.