1. In this problem we will consider the limitations of the linear classifier.

   **a)** Consider the set of binary classification problems, where the observation variable \( X = (X_1, X_2) \) is binary, i.e. \( X \in \{0,1\} \times \{0,1\} \), and non-degenerate, i.e. all possible configurations \((X_1, X_2)\) have non-zero probability. Consider the decision function
   
   \[
   h(x) = \begin{cases} 
   1, & w^T x + b \geq 0 \\
   0, & w^T x + b < 0 
   \end{cases}
   \]

   and let the class label \( Y \in \{0,1\} \) be a deterministic function of \( X \), \( Y = f(X) \). Show that there is at least one binary mapping \( f : \{0,1\} \times \{0,1\} \to \{0,1\} \) such that the observation \( X \) cannot be classified with zero probability of error with the decision function \( h(x) \). Given that, by making \( Y = f(x) \), it is easy to obtain zero probability of error on this problem, this observation is clearly problematic. Compute the smallest probability of error achievable with your function \( f(\cdot) \), when all configurations of \( X \) are equally likely.

   **b)** The extension of the linear classifier to problems with \( M \) classes is to define a set of functions
   
   \[
   g_i(x) = w_i^T x + b_i, \; i = 1, \ldots, M
   \]

   and assign \( x \) to class \( i \) if \( x \in R_i \) where
   
   \[
   R_i = \{x | g_i(x) \geq g_j(x) \forall j \neq i\}.
   \]

   Show that this classifier can only implement convex decision regions by showing that if \( x_1 \in R_i \) and \( x_2 \in R_i \) then \( \lambda x_1 + (1 - \lambda) x_2 \in R_i \), \( \forall 0 \leq \lambda \leq 1 \).

2. Let \( g(x) = w^T x + b \) and consider the hyperplane \( g(x) = 0 \).

   **a)** show that the Euclidean distance from a point \( x_a \) to the hyperplane is \( |g(x_a)|/||w|| \) by minimizing \( ||x - x_a||^2 \) subject to \( g(x) = 0 \).

   **b)** show that the projection of \( x_a \) onto the hyperplane is
   
   \[
   x_p = x_a - \frac{g(x_a)}{||w||^2} w.
   \]
3. Given a training set \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) the linear regression problem consists of finding the parameters \( \mathbf{a} = (w, b) \) of the function \( f(x) = w^T x + b \)

which minimize the empirical risk

\[
L(w, b) = \sum_{i=1}^{n}(y_i - w^T x_i - b)^2.
\]

a) show that the optimal solution is of the form

\[
\mathbf{a} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}
\]

and determine how the matrix \( \mathbf{X} \) and vector \( \mathbf{y} \) depend on the training patterns.

b) the matrix inversion above can be too expensive for some applications. Derive a stochastic gradient descent algorithm for this problem, i.e. an algorithm to similar the one we have studied in class for the Perceptron.

c) We have discussed in class that it is usually a good idea to penalize “complicated solutions”. In the regression problem, this can be accomplished by minimizing the alternative empirical risk

\[
L(w, b) = \sum_{i=1}^{n}(y_i - w^T x_i - b)^2 + \lambda w^T w,
\]

resulting in what is usually called ridge regression. The second term favors solutions that make \( w \) close to zero therefore penalizing those that have many degrees of freedom. Show that the optimal solution can be written as

\[
\mathbf{w}^* = \sum_i \alpha_i \mathbf{x}_i
\]

and use this to re-write \( L(w, b) \) in a form that only depends on the training patterns \( \mathbf{x}_i \) through their inner products. Solve the resulting problem to obtain the function

\[
y = f(\mathbf{x})
\]

in a form that only depends on the training patterns through their inner products with \( \mathbf{x} \).
4. We have seen in class that, for a training set \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) the Perceptron algorithm can be implemented as follows.

- set \( w_0 = 0, b_0 = 0, k = 0, R = \max_i ||x_i|| \)
- repeat
  - for \( i = 1, \ldots, n \)
    * if \( y_i(w_k^T x_i + b_k) \leq 0 \) then
      · set \( w_{k+1} = w_k + \eta y_i x_i \)
      · set \( b_{k+1} = b_k + \eta y_i R^2 \)
      · set \( k = k + 1 \)
- until there are no classification errors

a) Is the learning rate \( \eta \) relevant? Why?

b) show that an equivalent algorithm is

- set \( \alpha_0 = 0, b = 0, R = \max_i ||x_i|| \)
- repeat
  - for \( i = 1, \ldots, n \)
    * if \( y_i(\sum_{j=1}^n \alpha_j y_j x_j^T x_i + b) \leq 0 \) then
      · set \( \alpha_{i+1} = \alpha_i + 1 \)
      · set \( b = b + y_i R^2 \)
- until there are no classification errors

c) Can you give an interpretation to the parameters \( \alpha_i \)? Which among the samples \( x_i \) are the hardest to classify?

d) One of the interesting properties of this implementation is that it only depends on the dot-products \( x_i^T x_j \). Why is this interesting? Can you re-write the decision function

\[
h(x) = sgn(w^T x + b)
\]

in this form?
5. Let $K_1$ and $K_2$ be two dot-product kernels on $X \times X$, $X \subseteq \mathbb{R}^n$, $a \geq 0$, $f : X \rightarrow \mathbb{R}$, $\phi : X \rightarrow \mathbb{R}^m$, $K_3$ a dot-product kernel on $\mathbb{R}^m \times \mathbb{R}^m$, and $B$ a positive definite matrix. Show that the following functions are dot-product kernels.

1. $K(x, z) = K_1(x, z) + K_2(x, z)$.
2. $K(x, z) = a K_1(x, z)$.
3. $K(x, z) = K_1(x, z) K_2(x, z)$.
4. $K(x, z) = f(x) f(z)$.
5. $K(x, z) = x^T B z$.

6. Many of the classical linear techniques, such as PCA or Fisher’s Linear Discriminant (FLD), can be made non-linear by application of the “kernel trick”. Here we look at extensions to FLD. Given two training sets, $D_1$ from class 1 and $D_2$ from class 2, the FLD is the vector that maximizes

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

where

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

$$S_W = \sum_{i=1}^{2} \sum_{x \in D_i} (x - m_i)(x - m_i)^T$$

$$m_i = \frac{1}{|D_i|} \sum_{x \in D_i} x$$

The FLD is known to be optimal only when the classes are Gaussian and have equal covariance, a situation that rarely holds in practice. One possibility for obtaining a non-linear extension is to 1) apply a transformation $\Phi$ to a feature space, and 2) compute the FLD on that feature space. Clearly, this will map into a non-linear discriminant in the original space. The extension is straightforward, i.e.

$$J(w) = \frac{w^T S_B^\Phi w}{w^T S_W^\Phi w}$$

where

$$S_B^\Phi = (m_1^\Phi - m_2^\Phi)(m_1^\Phi - m_2^\Phi)^T$$

$$S_W^\Phi = \sum_{i=1}^{2} \sum_{x \in D_i} (\Phi(x) - m_i^\Phi)(\Phi(x) - m_i^\Phi)^T$$

$$m_i^\Phi = \frac{1}{|D_i|} \sum_{x \in D_i} \Phi(x)$$

but the computation is intractable when $\Phi(x)$ is very high (e.g. infinite) dimensional. To “kernalize” this optimization problem, we simply need to express the cost as a function that only depends on dot-products of the input patterns. Show that, under the assumption that $w$ can be written as

$$w = \sum_{x_i \in D_1 \cup D_2} \alpha_i \Phi(x_i),$$
the cost can be expressed as

\[ J(\alpha) = \frac{\alpha^T M\alpha}{\alpha^T N\alpha} \]

where \( \alpha = (\alpha_1, \ldots, \alpha_{|D_1 \cup D_2|}) \) and the matrices \( M \) and \( N \) only depend on the \( x_i \) through a suitably defined kernel function \( k(\cdot, \cdot) \).