Exercise Sheet 6

Notes on Dual Problem, KKT-Optimality and Slater's Condition

Consider an optimization problem in the **canonical** form:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f_0(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leqslant 0, \quad i = 1, ..., m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, ..., p \end{aligned}$$

The **Lagrange function** $\mathcal{L} \colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as a weighted sum of the objective and constraint functions:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \mu_i h_i(\boldsymbol{x}),$$

where $\boldsymbol{x}$ is called **primal** and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ the **dual** variables.

The (Lagrange) **dual function** $g \colon \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as:

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x} \in \text{domain}(f_0)} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}).$$

The convex optimization problem

$$\begin{aligned} \underset{(\boldsymbol{\lambda}, \boldsymbol{\mu})}{\text{maximize}} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \succeq \mathbf{0} \end{aligned}$$

is called the (Lagrange) **dual problem**.

In the Lagrange optimization framework the KKT-conditions are used to find the primal and dual optimal solutions.

Theorem 1 (Optimality Conditions) *For any optimization problem with differentiable objective and constraint functions for which strong duality obtains, any pair of primal and dual optimal* $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ *must satisfy KKT-conditions:*

$$\begin{aligned} &\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0, && \textit{(stationarity)} \\ &f_i(\boldsymbol{x}^*) \leqslant 0, && \textit{(primal feasibility)} \\ &h_i(\boldsymbol{x}^*) = 0, && \textit{(primal feasibility)} \\ &\lambda_i^* \geqslant 0, && \textit{(dual feasibility)} \\ &\lambda_i^* \cdot f_i(\boldsymbol{x}^*) = 0 && \textit{(complementary slackness)} \end{aligned}$$

For any convex problem, the KKT-conditions are sufficient for $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ to be optimal with zero duality gap.

Definition 2 (Slater's Condition) *We say that a convex optimization problem*

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f_0(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leqslant 0, \quad i = 1, ..., m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, ..., p \end{aligned}$$

satisfies the Slater's condition if there exists a strictly feasible point $\boldsymbol{x}$, i.e., $f_i(\boldsymbol{x}) < 0$ and $h_j(\boldsymbol{x}) = 0$ for all i, j.

Theorem 3 (Slater's Theorem) *For any convex problem for which Slater's condition holds, the KKT-conditions provide necessary and sufficient condition for $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ to be primal and dual optimal with zero duality gap.*

Exercise 1: Dual formulation of the Soft-Margin SVM

(a) Soft-margin SVM is given by a convex optimization problem: the objective is convex and the inequality constraints are linear (therefore also convex). Furthermore, the Slater's Theorem guarantees that if there is a feasible point $(\boldsymbol{w}, b, \boldsymbol{\xi})$ which strictly satisfies the inequality constraints, then strong duality holds. Here, for any $(\boldsymbol{w}, b)$ we can always choose sufficiently large values for the slack variables $\boldsymbol{\xi}$ such that all inequality constraints are strictly satisfied. Therefore, strong duality (in contrast to the hard-margin) holds always for the soft-margin formulation.

(b) First we rewrite the optimization problem in the canonical form:

$$\begin{aligned} \underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & 1 \; - \; \xi_i - y_i(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + b) \leqslant 0, \quad i = 1, ..., n \\ & - \xi_i \leqslant 0, \qquad\qquad\qquad\qquad\qquad i = 1, ..., n \end{aligned}$$

The Lagrangian is given as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i(1 - \xi_i - y_i(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + b)) + \sum_{i=1}^{n} \beta_i(-\xi_i),$$

where $(\boldsymbol{w}, b, \boldsymbol{\xi})$ are the primal and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are the dual variables. The corresponding dual function is given as

$$\begin{aligned}
g(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i(1 - \xi_i - y_i(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + b)) + \sum_{i=1}^n \beta_i(-\xi_i) \\
&= \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^n \alpha_i y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right) + \left(-\sum_{i=1}^n \alpha_i y_i b\right) + \left(C\sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \xi_i - \sum_{i=1}^n \beta_i \xi_i\right) + \sum_{i=1}^n \alpha_i \\
&= \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^n \alpha_i y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right) + \left(-b\sum_{i=1}^n \alpha_i y_i\right) + \left(\sum_{i=1}^n \xi_i(C - \alpha_i - \beta_i)\right) + \sum_{i=1}^n \alpha_i \\
&= \inf_{\boldsymbol{w}} \left\{\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^n \alpha_i y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right\} + \inf_b \left\{-b\sum_{i=1}^n \alpha_i y_i\right\} + \inf_{\boldsymbol{\xi}} \left\{\sum_{i=1}^n \xi_i(C - \alpha_i - \beta_i)\right\} + \sum_{i=1}^n \alpha_i
\end{aligned}$$

Note that the minimization over b and $\boldsymbol{\xi}$ is completely unrestricted. Therefore, the only way for the infimum to be bigger that $-\infty$ if the constrains $\sum_{i=1}^n \alpha_i y_i = 0$ and $C - \alpha_i - \beta_i = 0$ are satisfied. This is in agreement with the results below. To find the minimizing arguments $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*)$ we set the gradient of the corresponding terms to zero as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{w}} \mathcal{L} = \boldsymbol{w} - \sum_{i=1}^n \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) = \mathbf{0} \qquad &\Longrightarrow \qquad \boldsymbol{w}^* = \sum_{i=1}^n \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) \\
\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i = 0 \qquad &\Longrightarrow \qquad \sum_{i=1}^n \alpha_i y_i = 0 \\
\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \qquad &\overset{\beta_i \geqslant 0}{\Longrightarrow} \qquad 0 \leqslant \alpha_i \leqslant C
\end{aligned}$$

We now eliminate $\boldsymbol{w}$ by inserting the value $\boldsymbol{w}^*$ back into the equation and respecting the corresponding constrains:

$$\begin{aligned}
g(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \inf_{\boldsymbol{w}} \left\{\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^n \alpha_i y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\right\} + \inf_b \left\{-b\sum_{i=1}^n \alpha_i y_i\right\} + \inf_{\boldsymbol{\xi}} \left\{\sum_{i=1}^n \xi_i(C - \alpha_i - \beta_i)\right\} + \sum_{i=1}^n \alpha_i \\
&= \frac{1}{2}\|\boldsymbol{w}^*\|^2 - \sum_{i=1}^n \alpha_i y_i {\boldsymbol{w}^*}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + \inf_b \left\{-b\sum_{i=1}^n \alpha_i y_i\right\} + \inf_{\boldsymbol{\xi}} \left\{\sum_{i=1}^n \xi_i(C - \alpha_i - \beta_i)\right\} + \sum_{i=1}^n \alpha_i \\
&= \begin{cases} -\frac{1}{2}\sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) + \sum_{i=1}^n \alpha_i, & \text{if } \sum_{i=1}^n \alpha_i y_i = 0 \text{ and } C - \alpha_i - \beta_i = 0 \\ -\infty, & \text{otherwise} \end{cases}
\end{aligned}$$

where in the last equation we used

$$\begin{aligned}
\frac{1}{2}\|\boldsymbol{w}^*\|^2 - \sum_{i=1}^n \alpha_i y_i {\boldsymbol{w}^*}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) &= \frac{1}{2}\langle \sum_{i=1}^n \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i), \sum_{i=1}^n \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle - \sum_{i=1}^n \alpha_i y_i \langle \sum_{j=1}^n \alpha_j y_j \boldsymbol{\phi}(\boldsymbol{x}_j), \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle \\
&= \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) - \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) \\
&= -\frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)
\end{aligned}$$

as well as

$$\inf_b \left\{ -b \sum_{i=1}^{n} \alpha_i y_i \right\} = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} \alpha_i y_i = 0 \\ -\infty, & \text{otherwise} \end{cases}$$

and

$$\inf_{\boldsymbol{\xi}} \left\{ \sum_{i=1}^{n} \xi_i (C - \alpha_i - \beta_i) \right\} = \begin{cases} 0, & \text{if } C - \alpha_i - \beta_i = 0 \text{ for all } i \\ -\infty, & \text{otherwise.} \end{cases}$$

That is, the **dual function** is given as

$$\boxed{g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i, & \text{if } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } C - \alpha_i - \beta_i = 0 \\ -\infty, & \text{otherwise} \end{cases}}$$

Given an optimization problem in the canonical form, in the corresponding dual we maximize the dual function (here $g(\boldsymbol{\alpha}, \boldsymbol{\beta})$) subject to the non-negativity constraints on the dual variables (here $\boldsymbol{\alpha}, \boldsymbol{\beta} \succeq \mathbf{0}$). Observe that the dual function has implicit constraints. Since we aim to maximize g over the dual variables – we do not care about the case where $g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\infty$. Therefore, we can write the implicit constraints in the definition of g as explicit constrains of the optimization problem. This gives the the following formulation of the dual problem (for the soft-margin SVM):

$$\boxed{\begin{aligned} &\underset{\alpha_1, \ldots, \alpha_n}{\text{maximize}} && \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) \\ &\text{subject to} && \forall i \colon 0 \leqslant \alpha_i \leqslant C \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}}$$

In particular, from $C - \alpha_i - \beta_i = 0$ we get $C - \alpha_i = \beta_i$. That is, $\boldsymbol{\beta}$ is uniquely determined by the values of $\boldsymbol{\alpha}$ and is effectively eliminated from the above optimization problem. Furthermore, due to $\alpha_i, \beta_i \geqslant 0$ we get a box constraint $0 \leqslant \alpha_i \leqslant C$.

(c) From the previous solution in (b) we know that

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{\phi}(\boldsymbol{x}_i).$$

To find b^* we use the KKT condition (complementary slackness) "$\lambda_i \cdot f_i(\boldsymbol{x}) = 0$". Note that the data points $\boldsymbol{x}_i$ with $\alpha_i = 0$ do not contribute to the decision boundary. All other points with $\alpha_i > 0$ constitute the support vectors. Points with $\alpha_i = C$ lie inside the margin (or even on the wrong side of the decision boundary). Consider a support vector with $0 < \alpha_i < C$. Such support vectors lie exactly on the margin boundary! This follows from the complementary slackness:

$$\begin{aligned} \alpha_i \cdot (1 - \xi_i - y_i(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + b)) = 0 \quad &\overset{\alpha_i > 0}{\Longrightarrow} \quad b = y_i(1 - \xi_i) - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) \\ \beta_i(-\xi_i) = 0 \quad &\overset{\beta_i = C - \alpha_i > 0}{\Longrightarrow} \quad \xi_i = 0, \end{aligned}$$

which together implies

$$0 < \alpha_i < C \quad \Longrightarrow \quad b = y_i - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) = y_i - \sum_{j=1}^{n} \alpha_j y_j \boldsymbol{\phi}(\boldsymbol{x}_j)^\top \boldsymbol{\phi}(\boldsymbol{x}_i).$$

(d) By replacing $\boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ we get kernelized version:

$$\boxed{\begin{array}{ll} \underset{\alpha_1,\dots,\alpha_n}{\text{maximize}} & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} & \forall i\colon 0 \leqslant \alpha_i \leqslant C \text{ and } \sum_{i=1}^n \alpha_i y_i = 0. \end{array}}$$

The corresponding decision function is given as

$$f(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b) = \operatorname{sign}(\sum_{i=1}^n \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b)$$

Exercise 2: SVMs and Quadratic Programming

Consider first the objective

$$\underset{\alpha_1,\dots,\alpha_n}{\text{minimize}} \quad \frac{1}{2}\underbrace{\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)}_{=\boldsymbol{\alpha}^\top P \boldsymbol{\alpha}} - \underbrace{\sum_{i=1}^n \alpha_i}_{=\boldsymbol{1}^\top \boldsymbol{\alpha}}$$

where $P_{i,j} = y_i y_j K_{i,j}$ with K being the kernel matrix (reminder: $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$). That is,

$$\boxed{P = \operatorname{diag}(y_1, ..., y_n) \cdot K \cdot \operatorname{diag}(y_1, ..., y_n)},$$

where $y_1, ..., y_n$ are the labels of $x_1, ..., x_n$, respectively. Furthermore, $\boldsymbol{q} = -\boldsymbol{1} \in \mathbb{R}^n$, where $\boldsymbol{1} = (1, ..., 1)$ is a vector of one's. The inequality constraints $0 \leqslant \alpha_i \leqslant C$ can be represented as follows:

$$G = \begin{bmatrix} -I \\ I \end{bmatrix}, h = \begin{bmatrix} \boldsymbol{0} \\ C \cdot \boldsymbol{1} \end{bmatrix},$$

where $I \in \mathbb{R}^{n \times n}$ denotes the identity matrix, that is, $G \in \mathbb{R}^{2n \times n}$ and $h \in \mathbb{R}^{2n}$. The equality constraint $\sum_{i=1}^n \alpha_i y_i = 0$ can be represented as

$$A = \boldsymbol{y}^\top, b = 0,$$

where $\boldsymbol{y} = (y_1, ..., y_n)$.

```python
import numpy as np
import scipy, scipy.spatial
import cvxopt, cvxopt.solvers

def getGaussianKernel(X1, X2, scale):
    D = scipy.spatial.distance.cdist(X1,X2,'sqeuclidean')
    return np.exp(-D/(2*scale**2))

def getQPMatrices(K, Y, C):
    # Prepare matrices
    n = Y.shape[0]
    P = Y[:,np.newaxis]*K*Y[np.newaxis,:]
    # # alternatively
    # #-------------
    # diag = np.diag(Y)
    # P = np.matmul(diag, np.matmul(K, diag))
    # #-------------
    q = -np.ones([n])
    G = np.concatenate([-np.identity(n), np.identity(n)])
    h = np.concatenate([np.zeros([n]), C * np.ones([n])])
    A = np.reshape(Y, (1,n))
    b = np.array([0.0])

    # Convert to CVXOPT matrices
    P = cvxopt.matrix(P)
    q = cvxopt.matrix(q)
    G = cvxopt.matrix(G)
    h = cvxopt.matrix(h)
    A = cvxopt.matrix(A)
    b = cvxopt.matrix(b)

    return P,q,G,h,A,b

def getTheta(K, Y, alpha, C):
    # First we need to find a support vector with 0 < alpha_i < C.
    # Instead of looking at all possible alpha's in a loop, we use  the midpoint heuristic.
    # Note: the value lying closer to C / 2 is more likely to satify this condition.
    # Considering the absolute difference np.abs ensures that the value \alpha_i does
    # not lie to close to the boundaries 0 or C improving upon the numerical stability.
    sv = np.argmin(np.abs(alpha - C / 2.0))
    theta = Y[sv] - np.dot(K[sv,:], alpha * Y)
    return theta

def fit(self, X, Y):
    K = getGaussianKernel(X, X, self.scale)
    P, q, G, h, A, b = getQPMatrices(K, Y, self.C)

    alpha = np.array(cvxopt.solvers.qp(P, q, G, h, A, b)['x']).flatten()
    th = 1e-6 * alpha.mean()
    ind = alpha > th # determine (robust) support vectors (alternatively set th = 0)
    self.X, self.Y, self.alpha = X[ind], Y[ind], alpha[ind]

    self.theta = getTheta(K, Y, alpha, self.C)

def predict(self, X):
    K = getGaussianKernel(X, self.X, self.scale)
    Y = np.sign(np.dot(K, self.alpha * self.Y) + self.theta)
    return Y
```