(6) Practice Problems

(1) Solve the following dual SVM problem, $\min \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$ such that $\sum_{i=1}^{m+p} \alpha_i y_i = 0,\ \alpha \geq 0$ with projected gradient, where $Q = YXX^TY$, X is the data matrix (features) with size $n \times d$ and $Y = \mathrm{diag}(y)$ where y is the labels (-1 or 1) with size $n \times 1$. Run projected gradient for at most $1e4$ iterations or until you reach the accuracy of $\epsilon = 1e - 3$, whichever is first. Display the final objective value. Generate the data as follows:

MATLAB:

```
rng(123);
d = 50;
n = 2;
slope = rand(n,1);
intercept = 0;
X = randn(d,n);
y = ones(d, 1);
y(X*slope + intercept < 0) = -1;
alpha = zeros(d, 1);
tol = 1e-3;
maxiter = 1e4;
```

Python:

```
np.random.seed(123)
d = 50
n = 2
slope = np.random.rand(n)
intercept = 0
X = np.random.randn(d, n)
y = np.ones(d)
y[X.dot(slope) + intercept < 0] = -1
alpha = np.zeros(d)
tol = 1e-3
maxiter = int(1e4)
```

To compute the projection in **MATLAB** you can use `quadprog` as follows:

$$\min \frac{1}{2}x^T Px + q^T x, \quad \text{s.t. } Gx \le h,\ Ax = b,\ lb \le x \le ub$$

$$x = quadprog(P, q, G, h, A, b, lb, ub, x0, options)$$

```
options = optimoptions('quadprog', 'Display', 'off');
alpha = quadprog(eye(d), -alpha, [], [], y', 0, zeros(d, 1), [], [], options);
```

To compute the projection in **Python** use `qpsolvers` as follows:

$$x = solve\_qp(P, q, G, h, A, b, lb, ub, solver = \text{"}cvxopt\text{"})$$

```
!pip install qpsolvers
import numpy as np
from qpsolvers import solve_qp
P = np.eye(d)
q = -alpha
G = None
h = None
A = y.T
b = np.array([0.0])
lb = np.zeros(d)
ub = None
alpha = solve_qp(P, q, G, h, A, b, lb, ub, solver="cvxopt")
```

Slope and Intercept can be found as follows:

MATLAB:

```
w = X'*(y.*alpha);
[~,ind] = max(alpha);
beta = y(ind) - X(ind,:)*w;
```

Python:

```
w = X.T @ (y * alpha)
ind = np.argmax(alpha)
beta = y[ind] - X[ind, :]  @ w
```