Machine Learning Homework 10

Authors: Youran Wang, Tian Hao, Jesus Arturo Sol Navarro, Boris Petreski

1 Problem 6

- Similarities
  - Both are supervised learning algorithms.
  - Both algorithms are meant for binary classification, but SVM can be extended to multiclass problems using techniques like one-vs-one or one-vs-rest.
- Diferences
  - Perceptron tries to find any hyperplane that separates the data. If the data is linearly separable, it converges; if not ,it fails. On the other hand, SVM finds the optimal hyperplane, maximizing the margin between the hyperplane and the closest data points.
  - Perceptron can only handle linearly separable data. SVM can handle non-linearly separable data by using kernel functions.
  - Perceptron uses a simple update rule to adjust weights. SVM solution comes from a convex optimization problem.

2 Problem 7

The connection between soft-margin SVM and Logistic Regression lies in their shared goal of linear classification. Both solutions are derived from convex optimization problems . Both aim to find a decision boundary , but they use different loss functions: hinge loss for SVM and log-loss for Logistic Regression.

3 Problem 8

- a)
  $Q = (yy^T) \circ (XX^T)$
- b)
  - Q is positive semi-definite.
    - XX^T is positive semidefinite from the dot product.
    - Scaing by yy^T preserves positive-definiteness.
  - The dual function is concave as it is the sum of a concave quadratic term $\frac{1}{2}\alpha^T Q\alpha$ and a linear term $\alpha^T \mathbf{1}_N$.
  - For concave functions, any local minimum is a global maximum.Therefore,the local maximizer of $g(\alpha)$ is the global minimum.

4 Problem 9

The inequality $\epsilon \leq \frac{s}{N}$ holds because, for hard-margin SVM, only support vectors have the potential to be misclassified during LOOCV. Removing a non-support vector does not affect the decision boundary. The fraction of potentially misclassified samples during LOOCV, represented by ϵ, is bounded by the ratio $\frac{s}{N}$.

5 Problem 10

5.1 Task 1: Solving the SVM dual problem

```python
def solve_dual_svm(X, y):
    """Solve the dual formulation of the SVM problem.

    Parameters
    ----------
    X : array, shape [N, D]
        Input features.
    y : array, shape [N]
        Binary class labels (in {-1, 1} format).

    Returns
    -------
    alphas : array, shape [N]
        Solution of the dual problem.
    """
    ### TODO: Your code below ###
    data_dim= X.shape[0]
    # These variables have to be of type cvxopt.matrix
    #Gram Matrix

    gram= np.dot(X,X.T)
    Y= np.diag(y)
    P_numpy=np.dot(Y, np.dot(gram, Y))
    P = matrix(P_numpy)

    q_numpy = -np.ones((data_dim, 1))
    q = matrix(q_numpy)

    G_numpy = -np.eye(data_dim)
    G = matrix(G_numpy)

    h_numpy = np.zeros((data_dim, 1))
    h = matrix(h_numpy)

    yyy=y.reshape(-1, 1)
    A_numpy= yyy.T
    A = matrix(A_numpy)

    b_numpy= np.array([[0]], dtype=np.float64)
    b = matrix(b_numpy)

    solvers.options['show_progress'] = False
    solution = solvers.qp(P, q, G, h, A, b)
    alphas = np.array(solution['x'])
    return alphas.reshape(-1)
```

5.2 Task 2: Recovering the weights and the bias

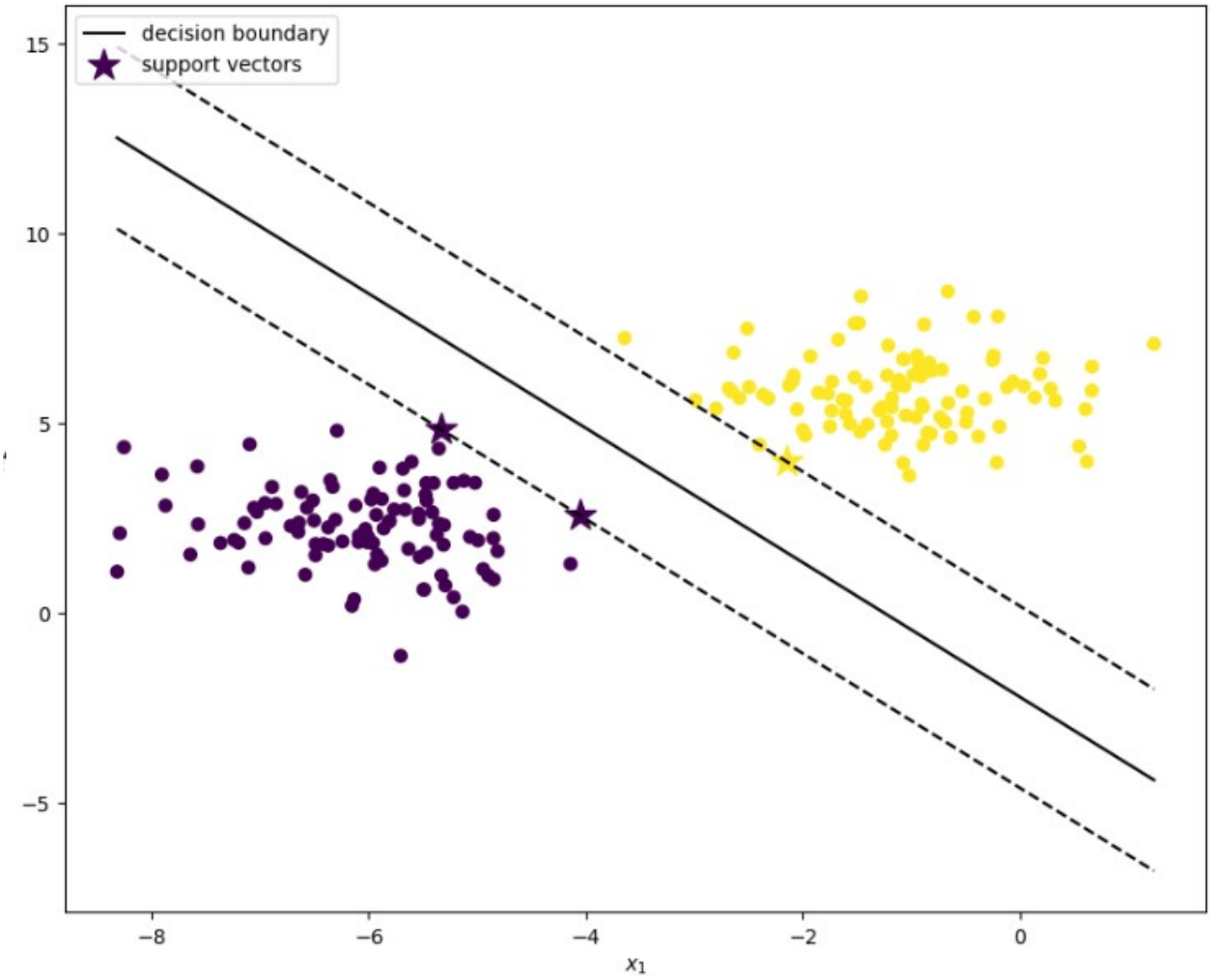
```python
def compute_weights_and_bias(alpha, X, y):
    """Recover the weights w and the bias b using the dual solution alpha.

    Parameters
    ----------
    alpha : array, shape [N]
        Solution of the dual problem.
    X : array, shape [N, D]
        Input features.
    y : array, shape [N]
        Binary class labels (in {-1, 1} format).

    Returns
    -------
    w : array, shape [D]
        Weight vector.
    b : float
        Bias term.
    """
    ### TODO: Your code below ###
    w = X.T @ (alpha * y)

    tolerance = 1e-3
    significant_indices = np.where(np.abs(alpha) > tolerance)[0]
```

```
    #print(significant_indices)
    b1 = y[significant_indices[0]] - X[significant_indices[0]] @ w
    b2 = y[significant_indices[1]] - X[significant_indices[1]] @ w
    b3 = y[significant_indices[2]] - X[significant_indices[2]] @ w
    b= (b1+b2+b3)/3
    return w, b
```

5.3 Visualize the result

```
alpha = solve_dual_svm(X, y)
w, b = compute_weights_and_bias(alpha, X, y)
print("w =", w)
print("b =", b)
print("support vectors:", np.arange(len(alpha))[alpha > alpha_tol])
```

```
w = [0.73935606 0.41780426]
b = 0.9199371454171242
support vectors: [ 78 134 158]
```



6 Problem 11

- The dot product $x_1^T x_2$ is a valid kernel.
- $(x_1^T x_2)^i$ is a valid kernel for any $i \geq 1$.
- Each term $a_i(x_1^T x_2)^i$ is a valid kernel since $a_i \geq 0$.

- The summation is valid because is a linear combination of valid kernels.
- Adding a constant term $a_0 \geq 0$ to the result of the summation preserves validity

Only kernel preserving opeations are used, thus k is a valid kernel.

7 Problem 12

- Taylor expansion of kernel

$$k(x_1, x_2) = \frac{1}{1 - x_1 x_2} = \sum_{n=0}^{\infty} (x_1 x_2)^n = \sum_{n=0}^{\infty} x_1^n x_2^n.$$

- Feature Transformation
  The taylor expansion implies that the dot product in the original sapce corresponds to a dot product in the transformed space where features are powers of x.

$$\phi(x) = [1, x, x^2, x^3, \dots]$$