DASC521 20/12/2021

Introduction to Machine Learning

Melih Can Yardı | 0078611

Homework 04 – One-Versus-All Support Vector Classification

Importing Data

- After importing NumPy, Pandas, Matplotlib, CVXOPT libraries, distance function from the SciPy library, and accuracy_score metric from Scikit-Learn, I've imported the images and labels datasets using Pandas' read_csv() function.

Train-Test Split

- Then, I have divided data into X_train, y_train, X_test, and y_test sets. I also created an N_train and N_test variables to use later.

```
X_train = images[:1000]
y_train = np.array(labels[:1000][0])
X_test = images[1000:]
y_test = np.array(labels[1000:][0])
```

```
N_train = len(y_train)
N_test = len(y_test)
```

Distance and Kernel Functions (Gaussian Kernel)

- Then, I have defined the gaussian_kernel function, using the code provided in LAB 08 with the following formulas:

Distance and Kernel Functions (Gaussian Kernel)

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2 = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)} = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2}$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{2s^2} \right)$$

```
def gaussian_kernel(X1, X2, s):
    D = dt.cdist(X1, X2)
    K = np.exp(-D**2 / (2 * s**2))
    return(K)
```

Learning Algorithm

- Then, I have defined one_vs_all_classification and predict functions; the former is the learning algorithm that separates multiple classes using support vector machines, and the latter makes the prediction by looking at the maximum of each class's predictions.

```
def one_vs_all_classification(C, class_num):
    y_train_ = y_train.copy()
    y_train_[y_train_ != class_num] = -1
    y_train_[y_train_ == class_num] = 1
    s = 10
    K_train = gaussian_kernel(X_train, X_train, s)
    yyK = np.matmul(y_train_[:,None], y_train_[None,:]) * K_train

    C = C
    epsilon = 1e-3
    print(f"C = {C}, class = {class_num}")

    P = cvx.matrix(yyK)
    q = cvx.matrix(-np.ones((N_train, 1)))
    G = cvx.matrix(np.vstack((-np.eye(N_train), np.eye(N_train))))
    h = cvx.matrix(np.vstack((np.zeros((N_train, 1)), C * np.ones((N_train, 1)))))
    A = cvx.matrix(1.0 * y_train_[None,:])
    b = cvx.matrix(0.0)

    result = cvx.solvers.qp(P, q, G, h, A, b)
    alpha = np.reshape(result["x"], N_train)
    alpha[alpha < C * epsilon] = 0
    alpha[alpha > C * (1 - epsilon)] = C

    support_indices, = np.where(alpha != 0)
    active_indices, = np.where(np.logical_and(alpha != 0, alpha < C))
    w0 = np.mean(y_train_[active_indices] * (1 - np.matmul(yyK[np.ix_(active_indices, support_indices)], alpha[support_indices])

    f_predicted = np.matmul(K_train, y_train_[:,None] * alpha[:,None]) + w0

    return f_predicted
```

```
def predict(C):
    f_predicted_1 = one_vs_all_classification(C=C, class_num=1)
    f_predicted_2 = one_vs_all_classification(C=C, class_num=2)
    f_predicted_3 = one_vs_all_classification(C=C, class_num=3)
    f_predicted_4 = one_vs_all_classification(C=C, class_num=4)
    f_predicted_5 = one_vs_all_classification(C=C, class_num=5)

    y_predicted = []

    for i in range(1000):
        predictions = [f_predicted_1[i], f_predicted_2[i], f_predicted_3[i], f_predicted_4[i], f_predicted_5[i]]
        max_value = max(predictions)
        max_index = predictions.index(max_value)
        y_predicted.append(max_index + 1)

    return y_predicted
```

- Using these two functions, I first made predictions with C parameter set to 10. Here is the resulting confusion matrix:

y_train	1	2	3	4	5
y_predicted					
1	207	1	0	9	0
2	2	199	1	1	0
3	0	1	204	6	0
4	0	1	4	185	1
5	0	0	0	0	178

- By setting C parameters to values 0.1, 1, 10, 100, and 1000, I have re-run the learning algorithm and saved the prediction accuracies.

```
train_predictions = []
for C in [0.1, 1, 10, 100, 1000]:
    train_predictions.append(predict(C))
```

```
train_accuracies = []

for prediction in train_predictions:
    train_accuracies.append(accuracy_score(y_train, prediction)),
```

- Then, I did the same steps for the test set, saved its prediction accuracies with different C parameters.

Visualization

- Lastly, I have visualized the relationship between the regularization parameter (C) and accuracy scores.