인공지능

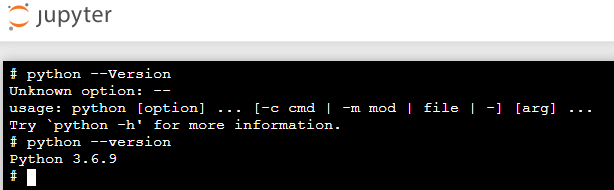
(단층 퍼셉트론)

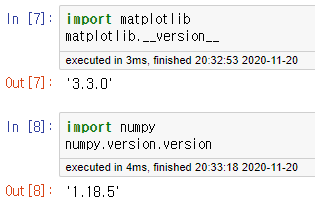
-점 클래스 분류-

20164091

송희령

0 > 실행 환경





연구실 컴퓨터의 Jupyter Notebook에서 실행하였습니다.

이번 실습에서 Numpy와 Matplotlib를 사용하였으며

파이썬은 3.6.9 버전을 이용하였습니다.

1 > 학습 데이터 코드

import matplotlib.pyplot as plt

import numpy as np

좌표의 시각화를 위해 matplotlib를 사용하여 점들을 표현하였으며 좌표데이터 구현을 위해 numpy를 사용하였습니다.

def ex\_calculate(input):

    global weights

    global bias

    activation = bias

    for i in range(2):

        activation += weights[i] \* input[i]

    if activation >= 0.0:

        return 1.0

    else:

        return 0.0

def ex\_train\_weights(X, Y, l\_rate, n\_epoch):

    global weights

    global bias

    for epoch in range(n\_epoch):

        sum\_error = 0.0

        for row, target in zip(X, Y):

            actual = ex\_calculate(row)

            error = target - actual

            bias = bias + l\_rate \* error

            sum\_error += error\*\*2

            for i in range(2):

                weights[i] = weights[i] + l\_rate \* error \* row[i]

            print(weights, bias)

        print("에포크 번호 = %d, 학습률 = %.3f, 오류 = %.3f" % (epoch, l\_rate, sum\_error))

    return weights

수업시간에 배운 계산과 가중치 학습 함수 입니다. 좌표 계산 역시 입력한 좌표들을 두 클래스로 분류하는것에 같은 맥락이므로 다른 수정 없이 사용하였습니다. Blue의 경우 계산 결고가가 0 미만일 경우, Red일 경우 0 이상일 경우로 설정하였습니다.

X = np.array([[0, 0], [-1, 2], [1, 3], [1, 1], [3, 1], [3, -1], [-3, 1], [-3, -1], [-1, -1], [-1, -3], [1, -2]])

Y = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

weights = [0.0, 0.0]

bias = 1.0

l\_rate = 0.1

n\_epoch = 10

weights = ex\_train\_weights(X, Y, l\_rate, n\_epoch)

print(weights, bias)

plt.scatter(X[:6, 0], X[:6, 1], color='r')

plt.scatter(X[6:11, 0], X[6:11, 1], color='b', marker='x')

plt.xlabel("X")

plt.ylabel("Y")

plt.title("Train Point(Red, Blue)")

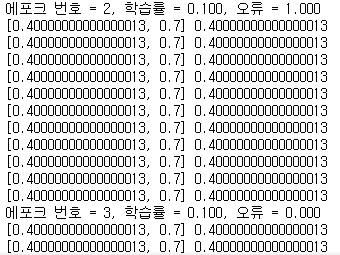
plt.show()

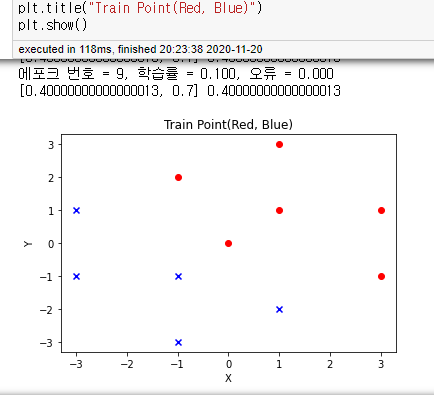
점 좌표들을 np.array를 통해 X에 저장하고, Y에는 해당 좌표들의 Red, Blue를 표현할 1과 0을 저장하였습니다.

가중치와 바이어스는 각각 0과 1로 설정하고 학습률은 0.1, 에포크는 10번 실행하도록 설정하고 학습 시작 하였습니다.

이후 학습 결과에 대한 점 좌표 표현을 위해 matplot의 scatter로 좌표들을 표현하였습니다.

2. 학습 실행 결과





실행 결과 중 4번째 실행에서 오류 0으로 잡히는 것을 보아 이때 학습이 완료된 것을 알 수 있습니다. 학습 결과 출력 이후 각 점들을 표현한 모습입니다.

3. 테스트 진행

# 좌표 입력 후 결과 출력

test\_X, test\_Y = input("두 점의 좌표를 입력해주세요 : ").split()

result = float(test\_X) \* weights[0] + float(test\_Y) \* weights[1] + bias

if(result > 0.0):

    print("빨간점 입니다. 결과값 : ", result)

else:

    print("파란점 입니다. 결과값 : ", result)

plt.scatter(X[:6, 0], X[:6, 1], color='red')

plt.scatter(X[6:11, 0], X[6:11, 1], color='b', marker='x')

plt.scatter(int(test\_X), int(test\_Y), color='y', marker='\*')

plt.xlabel("X")

plt.ylabel("Y")

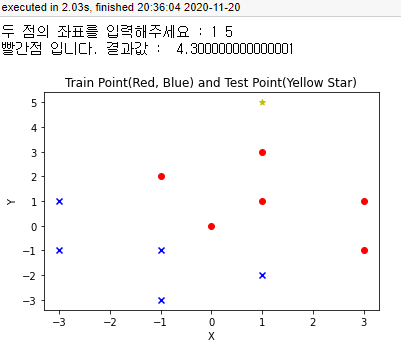
plt.title("Train Point(Red, Blue) and Test Point(Yellow Star)")

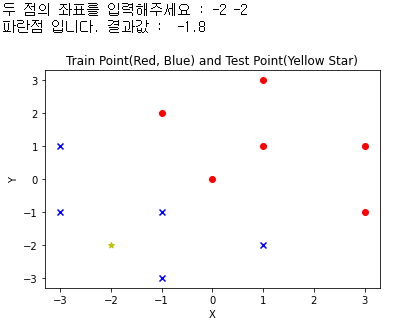
plt.show()

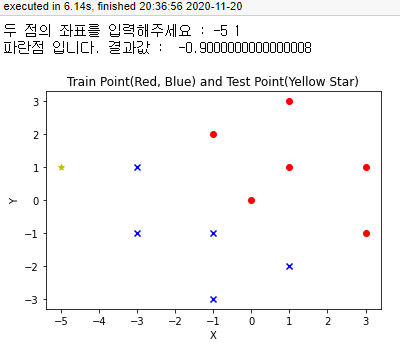
사용자로부터 좌표를 입력 받은 후 학습을 통해 나온 가중치값과 바이어스값을 통해 계산을 진행합니다.

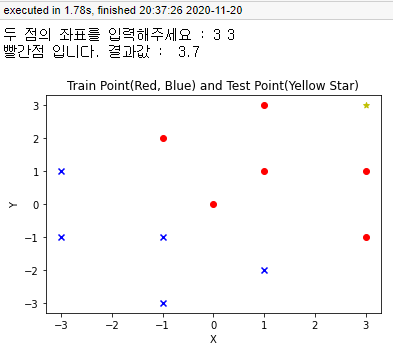
계산된 결과를 통해 해당 결과값이 0 미만이면 파란점, 0이상이면 빨간점이므로 각 결과에 맞는 텍스트 출력 후 사용자가 입력한 좌표값을 기존 학습 좌표에 대조해 보도록 합니다.

4. 테스트 진행 결과









해당 입력점(노란 별)에 대해 결과가 정상적으로 분류가 되는 것을 볼 수 있습니다.