Perceptrons - Making Predictions

Assuming that we have

```
weights = [-0.1, 0.206, -0.234]
```

```
def predict2(X, w):
    # bias = w[0]
    # activation = bias + w[1]* X[0] + w[2]* X[1]

    activation = w[0] + w[1]* X[0] + w[2]* X[1] # In "w" has 3 elements, In "X" has only 2 elemen
                                                # and w[0] means bias. (It means weight could be
    if activation >= 0.0:
        return 1.0
    else:
        return 0.0
```

```
# test predictions
dataset = [[2.7810836,2.550537003,0],
    [1.465489372,2.362125076,0],
    [3.396561688,4.400293529,0],
    [1.38807019,1.850220317,0],
    [3.06407232,3.005305973,0],
    [7.627531214,2.759262235,1],
    [5.332441248,2.088626775,1],
    [6.922596716,1.77106367,1],
    [8.675418651,-0.242068655,1],
    [7.673756466,3.508563011,1]]
```

```
bias=-0.1
w0=0.206
w1=-0.234
weights = [bias,w0,w1]
print(weights)
```

```
[-0.1, 0.206, -0.234]
```

```
weights = [-0.1, 0.206, -0.234] # 3 elements of "w"
```

여기서 데이터셋은 1행당 3개의 요소로 구성되어있어서 충분히 가중치(w)와 1대 1 대응이 가능해보이지만. predict2() 함수는 예측값을 알아보는 것이기 때문에 데이터셋의 세 번째(마지막) 요소는 정답을 가지고 있는 요소인것이다.

```
for row in dataset:
    prediction = predict2(row, weights)
    print("Expected={}, Predicted={}".format(row[-1], prediction))
```

```
Expected=0, Predicted=0.0
```

- More fancy !

```
Expected=0, Predicted=0.0
```

위 코드를 보면 prediction이라는 변수에 predict2() 함수의 연산을 거친 값을 받게 된다. 그리고 최종 출력문에는 기대치와 예상치를 출력한다. 이 때, 기대치는 정답 레이블이라고 생각하면 된다. row[-1]은 위의 데이터 셋에 있어서 마지막 값 즉, 정답 요소를 가리키며, 최종적으로 기대치와 예상치의 결과를 보고 정답률을 확인 할 수 있다.

```
Expected=1, Predicted=1.0
```

더블클릭 또는 Enter 키를 눌러 수정

```
def predict(row, weights):
    activation = weights[0]
    for i in range(len(row)-1):
        activation += weights[i + 1] * row[i]
    return 1.0 if activation >= 0.0 else 0.0


for row in dataset:
    prediction = predict(row, weights)
    print("Expected={}, Predicted={}".format(row[-1], prediction))
```


References

https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/

```
import matplotlib.pyplot as plt

plt.plot(3,4,'bo') # plot test
```

```
x = [1,2,3,4,5]

len(x)
```

```
y = []

for i in range(10):
  y.append(i*(3/4))

plt.plot(y)
for row in dataset:
  prediction = predict2(row, weights)
  answer = row[-1]
  if prediction == 0:
    plt.plot(row[0],row[1],'.', color = 'b')
  else:
    plt.plot(row[0],row[1],'.', color = 'r')

plt.show()
```

```
pred  = predict2([8,5],weights)
print(pred)
# 3.8은 0, 3.9는 1 (y = 3 기준)
# 3.9에 가까우면 1 (y = 3 기준)
# 3.89까진 0 (y = 3 기준)
# 3.899는 1 (y = 3 기준)

# 5.0은 0, 5 위로는 1 (y = 4 기준) # 이론1 : y >= 3/4 * x

# 12 < 3/4 * 9 이라서 0
# 12 >= 3/4 * 9 이라서 1
```

activation = w[0] + w[1]* X[0] + w[2]* X[1]