0807

2019년 8월 6일

지승훈

1 NumPy

다차원 배열과 행렬, 그리고 그것에 대한 여러 가지 연산을 지원하는 파이썬 라이브러리
일반적인 파이썬 연산에 비해 빠른 속도를 자랑함(파이썬 말고도 C언어로 짠 내부 구조)

```
In [ ]: import numpy as np
```

1.1 일반 파이썬과 속도 비교해보기

```
In [ ]: %%time
        ssum = 0
        for i in range(10000000):
            ssum += i
```

```
In [ ]: %%time
        ssum = sum([x for x in range(10000000)])
```

```
In [ ]: %%time
        ssum = np.sum(np.arange(10000000))
```

1.2 NumPy의 자료구조인 numpy.ndarray

```
In [ ]: # 기본적인 넘파이 배열 만들기
        a = np.array([1, 2, 3])
        print(a)
```

```
In [ ]: # 2x2 행렬 만들기
        a = np.array([[1, 2], [3, 4]])
        print(a)
```

```
In [ ]: # 일정 범위의 배열 만들기
        a = np.arange(10)
        print(a)
```

```
In [ ]: a = np.linspace(0, 10, 20)
        print(a)
```

```
In [ ]: # 배열의 형상과 차원, 자료형 알아보기
        a = np.arange(24).reshape(2, 3, 4)
        print(a, a.shape, a.ndim, a.dtype, sep='\n\n')
        # 참고: 배열의 차원은 가장 먼저 접근하는 인덱스의 차원일수록 고차원이다
```

```
In [ ]: # 모든 원소의 값이 0인 행렬 만들기
        a = np.zeros([3, 3]) # 배열의 형상을 입력하는 인수로는 튜플과 리스트 모두 사용 가능
        print(a)
```

```
In [ ]: # 모든 원소의 값이 1인 행렬 만들기
        a = np.ones([3, 3, 3])
        print(a)
```

```
In [ ]: # 지정한 범위 안의 무작위 정수로 이루어진 행렬 만들기
        a = np.random.randint(0, 10, (10, 10))
        print(a)
```

```
In [ ]: # 표준정규분포를 따르는 행렬 만들기
        a = np.random.randn(10, 3)
        print(a)
```

1.3 인덱싱과 슬라이싱

```
In [ ]: a = np.arange(36).reshape(6, 6)
        print(a)
```

```
In [ ]: a[2, 3] # == a[2][3]
```

```
In [ ]: a[2] # == a[2, :]
```

```
In [ ]: a[:, 2]
```

주의: NumPy의 인덱싱은 copy가 아닌 view이다.

```
In [ ]: a[:, 2] = 0
        print(a)
```

```
In [ ]: # fancy indexing
        a = np.arange(36).reshape(6, 6)
        print(a)
```

```
In [ ]: a[[2, 5]] # == a[[2, 5], :]
```

```
In [ ]: a[:, [3, 2]] # 인덱싱과 슬라이싱의 방법은 무궁무진하지만 여기까지!
```

1.4 행렬의 연산

```
In [ ]: # 행렬의 덧셈
        a = np.arange(10).reshape(2, 5)
        b = np.full((2, 5), 100)
        print(a, b, a + b, sep='\n\n')
```

```
In [ ]: # 행렬의 곱셈
        print(a * b)
```

```
In [ ]: #행렬의 모든 값 더하기
        a.sum()
```

```
In [ ]: a.sum(axis=0)
```

```
In [ ]: a.sum(axis=1)
```

```
In [ ]: # 행렬곱
        a = np.array([[1, 2], [3, 4]])
        b = np.array([[5, 6], [7, 8]])
        print(a, b, np.dot(a, b), sep='\n\n')
        print('*'*20)
        print(np.matmul(a, b))

        # np.dot과 np.matmul은 2차원 행렬에서는
        #계산 방식이 같지만 3차원 이상에서는 결과가 다르다.
```

```
In [ ]: # 행렬과 조건식
        a = np.random.randint(0, 10, 7)
        print(a)

In [ ]: a > 5

In [ ]: a[a > 5]
```

2 matplotlib

파이썬 자료의 시각화 라이브러리

NumPy, pandas 등과 함께 쓰이는 편

```
In [ ]: import matplotlib.pyplot as plt
        %matplotlib inline

In [ ]: #가장 간단한 선 그래프 만들어 보기
        plt.plot(np.sqrt(np.arange(5))) # 인수가 하나뿐이면 y값으로 넘김
        plt.title('a simple graph') # 그래프 이름
        plt.xlabel('x')
        plt.ylabel('y') # 각 축의 이름 적용
        plt.show()

In [ ]: x = np.linspace(-5, 5, 100)

        plt.plot(x, np.sin(x), color='green', label='sin(x)') # 그래프 색깔 적용
        plt.plot(x, np.cos(x), color='red', label='cos(x)')
        plt.title('graph of sin(x) and cos(x)')
        plt.xlabel('x')
        plt.ylabel('y')

        plt.grid(True) # 격자 적용
        plt.legend()
        plt.show()

In [ ]: # 여러 가지 line plot style
        x = np.arange(0, 5, 0.2)
```

```
plt.plot(x, x, 'r--', label='y=x')
plt.plot(x, x**2, 'bs', label='$y=x^2$')
plt.plot(x, x**3, 'g^', label='$y=x^3$')
plt.title('$y=x, y=x^2, y=x^3$')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc=2)
plt.show()
```

In [ ]:
```
# scatter plot
x = np.random.randn(500)
y = np.random.randn(500)

plt.scatter(x, y, color='000000', marker='x')

plt.xlim(-5, 5)
plt.ylim(-3, 3) # 각 축이 나타낼 범위 지정
plt.show()
```

In [ ]:
```
# 이미지 불러오기
image = plt.imread(r'C:\Users\USER\Pictures\Saved Pictures\icon_3+(1).gif') # 이미지를 넘
plt.imshow(image)
plt.axis('off') # 축 제거
plt.show()
```

In [ ]:
```
# 여러 개의 plot을 동시에 띄우기
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3)) # 사이즈는 직접 정해주지 않아도 됨
plt.subplot(1, 3, 1)
plt.suptitle('Categorical Plotting', x=0.5, y=1.1)
plt.bar(names, values)
plt.title('bar chart')
plt.subplot(132)
plt.scatter(names, values)
plt.title('scatter plot')
```

```
plt.subplot(133)
plt.plot(names, values)
plt.title('line plot')
plt.show()
```

3 NumPy와 matplotlib 적용해 보기

간단한 선형이진분류기를 학습시켜 보고, 그 과정을 시각화해 보자

```
In [ ]: from sklearn.datasets.samples_generator import make_blobs
```

```
In [ ]: # 데이터 준비
        XY, labels = make_blobs(n_samples=200, centers=2, random_state=4)
        labels[labels == 0] = -1
        data = np.hstack((XY, labels.reshape(200, 1)))

        dataNeg = data[data[:, 2] == -1]
        dataPos = data[data[:, 2] == 1]

        plt.scatter(dataNeg[:, 0], dataNeg[:, 1], label=-1, color='r')
        plt.scatter(dataPos[:, 0], dataPos[:, 1], label=1, color='b')
        plt.legend()
        plt.show()
```

```
In [ ]: # 가중치와 편향, 학습률 초기화
        W = np.array([-0.1, 0.1])
        b = np.array([0.3])

        lr = 0.001

        # 계단 함수 정의
        def stepFunc(matmul): # np.where() 함수로도 구현할 수 있으나 생략
            if matmul > 0:
                return 1
            else:
                return -1
```

In [ ]:
```
# 학습 셋 뽑기
trainSet = data[np.random.choice(len(data), 30)]
print(trainSet)

# 학습 횟수 초기화: 여기서는 반복문을 쓰지 않고 직접 일일이 확인해볼 것이기 때문에
# 별개의 블럭에 이터레이션 변수를 써놓고 따로 초기화함
i = 0
```

In [ ]:
```
# 학습시켜 보기
print('iteration:', i)
print('current weights:', W, b)

# i번째 표본의 값과 정답 레이블을 할당하고, 그래프 안에 그 위치를 표시함
xy = trainSet[i, :2]
answer = trainSet[i, 2]
plt.annotate(s='sample here', xy=(xy[0], xy[1]), xytext=(9, 7.5),
             arrowprops=dict(facecolor='k'))
print('train data:', xy)
xPlot = np.linspace(7, 12, 10) # line plot을 그리기 위한 설정
plt.plot(xPlot, (-W[0]/W[1])*xPlot-b/W[1], label='previous', color='gray') # 이전 그래프

# 행렬곱(여기서는 내적)으로 퍼셉트론 계산 수행, 퍼셉트론은 계단 함수로 분류값 판단
sigma = np.matmul(xy, W) + b
print('sigma:', sigma)
predict = stepFunc(sigma)

# 정답 레이블과 비교해 보고, 그 결과를 가중치 업데이트에 반영
print('predict:', predict, 'answer:', answer)
W += lr * (answer - predict) * xy
b += lr * (answer - predict)
print('updating values:', lr * (answer - predict) * xy, lr * (answer - predict))
print('updated weights:', W, b)

# 지금까지의 학습 결과를 시각화

plt.plot(xPlot, (-W[0]/W[1])*xPlot-b/W[1], label='updated', color='m')
```

```
        plt.scatter(dataNeg[:, 0], dataNeg[:, 1], label=-1, color='r')
        plt.scatter(dataPos[:, 0], dataPos[:, 1], label=1, color='b')
        plt.xlim(6.5, 12.5)
        plt.ylim(-2, 8)
        plt.legend(loc=1)
        plt.show()

        i += 1
```

In [ ]:
```
        # 한 에폭(주기)당 20번씩 학습하고,
        # 그 모델로 30번 테스트하고,
        # 모두 300에폭 반복했을 때 이 모델의 정답률 시각화
        epochs = 300
        sample = 50
        batch = 20
        W = np.array([-0.1, 0.1])
        b = np.array([0.3])
        lr = 0.001
        accuracyList = []

        for epoch in range(epochs):
            if epoch % 100 == 0:
                print('\n')
                print('-'*70)
                print('epoch:', epoch)
            mask = np.random.choice(len(data), sample)
            trainSet = data[mask[:batch]]
            testSet = data[mask[batch:]]
            correct = 0

            # 학습
            for trainIt in trainSet:
                xy = trainIt[:2]
                answer = trainIt[2]
                sigma = np.matmul(xy, W) + b
                predict = stepFunc(sigma)
```

```
            W += lr * (answer - predict) * xy
            b += lr * (answer - predict)
        print('weights learned:', W, 'bias learned:', b)

        # 테스트 후 정확도 평가
        for testIt in testSet:
            xy = testIt[:2]
            answer = testIt[2]
            sigma = np.matmul(xy, W) + b
            predict = stepFunc(sigma)
            if predict == answer:
                correct += 1
        accuracy = correct / len(testSet)
        print('accuracy:', accuracy)
        accuracyList.append(accuracy)

    # 시각화
    plt.figure(figsize=(20, 10))
    plt.plot(range(epochs), accuracyList)
    plt.title('Accuracy')
    plt.show()
```

4 더 알아보기

NumPy 도큐멘테이션 https://docs.scipy.org/doc/numpy/reference/index.html

Matplotlib https://matplotlib.org

갤러리 https://matplotlib.org/gallery/index.html

Matplotlib 기반 시각화 라이브러리인 seaborn https://seaborn.pydata.org/