데이터 과학

L11.1: kNN Practice

Kookmin University

모듈 불러오기

- 사용할 모듈 import 하기

```
import matplotlib.pyplot as plt
import random
from tqdm import tqdm
```

Iris dataset

- 아이리스(붓꽃) 데이터
  - 붓꽃 종류별로 꽃받침과 꽃잎의 길이 및 너비를 측정한 데이터

Iris Setosa

Iris Versicolour

Iris Virginica

Iris dataset

- 아이리스(붓꽃) 데이터
  - 붓꽃 종류별로 꽃받침과 꽃잎의 길이 및 너비를 측정한 데이터
  - https://archive.ics.uci.edu/ml/datasets/Iris

```
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
```

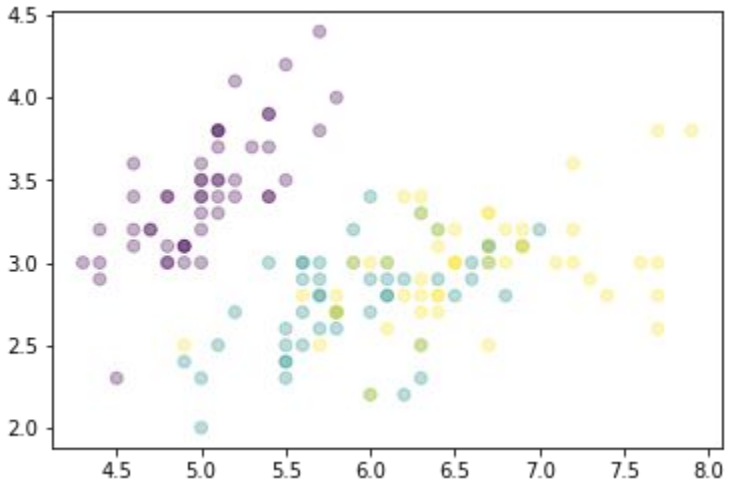
데이터 불러오기

```
data = []
with open('iris.data', 'r') as f:
   for line in f:
       if line.strip():
           item = line.strip().split(",")
           data.append( ([float(val) for val in item[:-1]], item[-1]) )

species = {s: i for i, s in enumerate(set(d[1] for d in data))}
data = [(d[0], species[d[1]]) for d in data]
```

데이터 살펴보기

```
plt.scatter([d[0][0] for d in data],
            [d[0][1] for d in data], c=[d[1] for d in data], alpha=0.3)
plt.show()
```



데이터 분리하기

- train data와 test data로 분리

```
random.shuffle(data)
train = data[:-30]
test = data[-30:]
```

distance()

```
def distance(a, b):
    s = 0
    for i in range(len(a[0])):
        s += (a[0][i] - b[0][i]) ** 2
    return s ** 0.5
```

knn_classify()

```
def knn_classify(k, query, train):
    knns = sorted((distance(point, query), point) for point in train)[:k]

    # 거리에따라 가중치를 주어 점수 계산
    scores = {}
    for dist, point in knns:
        scores[point[1]] = scores.get(point[1], 0) + 1/(1+dist)

    resp, score = max(scores.items(), key=lambda x: x[1])

    return resp
```

test 해보기

```
k = 3
correct = sum(1 for t in test if knn_classify(k, t, train) == t[1])
accuracy = correct/len(test)
print("accuracy:", accuracy)
```

최적의 k 찾기

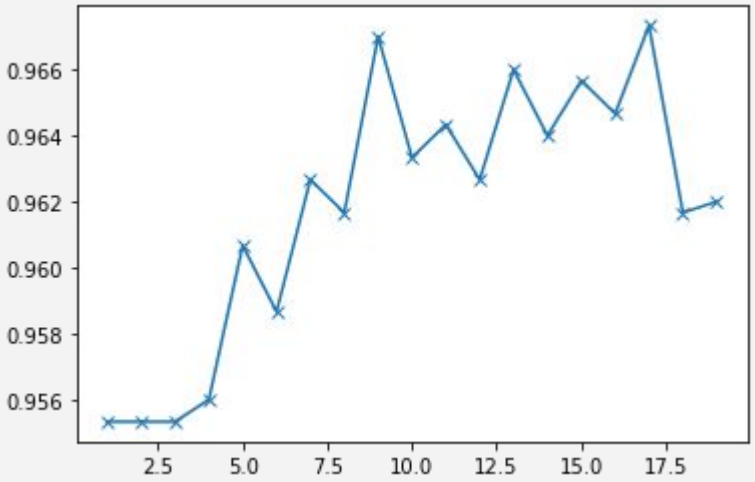
```
num_trials = 100
train_size = int(len(data) * 0.8)
test_size = len(data) - train_size

corrects = [0]*20

for i in tqdm(range(num_trials)):
    random.shuffle(data)
    train = data[:train_size]
    test = data[train_size:]

for k in range(1,20):
     corrects[k] += sum(1 for t in test if knn_classify(k, t, train) == t[1])
     corrects[k] /= num_trials * test_size

plt.plot(range(1,20), corrects[1:], '-x')
plt.show()
```



Questions?