AI Programming
[Week 14] Practice

2024. 12. 5.

목차

HW10 파일 준비

- main.py
- hw10 – data files
  - lin_test.txt
  - lin_train.txt
  - nonlin_test.txt
  - nonlin_train.txt

실습 목표

1. Linear Regression 실습

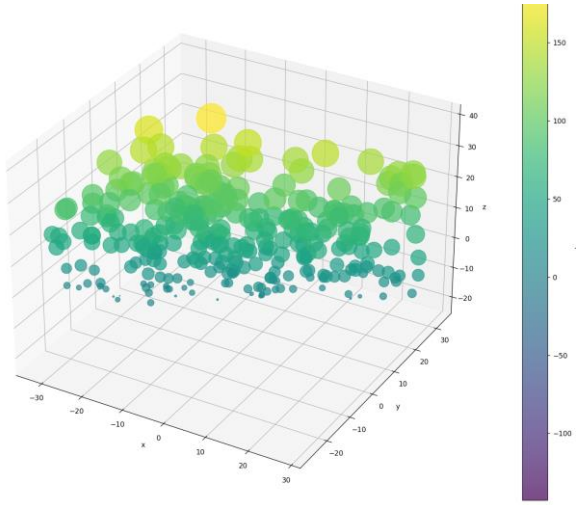
2. K-NN 구현 실습
- ML.kNN 구현
- ML.findCK 구현
- ML.updateCK 구현
- ML.sDistance 구현
- ML.takeAvg 구현

3. K-NN 실습

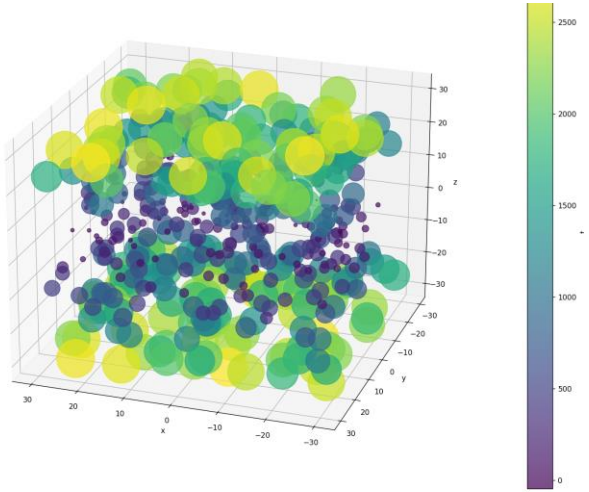
Linear Regression으로 non-linear, linear 문제 풀어보기 (5분)

```
Enter the file name of training data: lin_train.txt
Enter the file name of test data: lin_test.txt

Which learning algorithm do you want to use?
 1. Linear Regression
 2. k-NN
Enter the number: 1

RMSE:  0.3
```

```
Enter the file name of training data: nonlin_train.txt
Enter the file name of test data: nonlin_test.txt

Which learning algorithm do you want to use?
 1. Linear Regression
 2. k-NN
Enter the number: 1

RMSE:  82.33
```



lin_train.txt



nonlin_train.txt

K-NN 구현

ML.kNN(self, query)

query에 대해 가장 가까운 K개의 이웃을 찾아서
평균 값을 반환
- findCK, takeAvg 메소드 사용

ML.findCK(self, query)

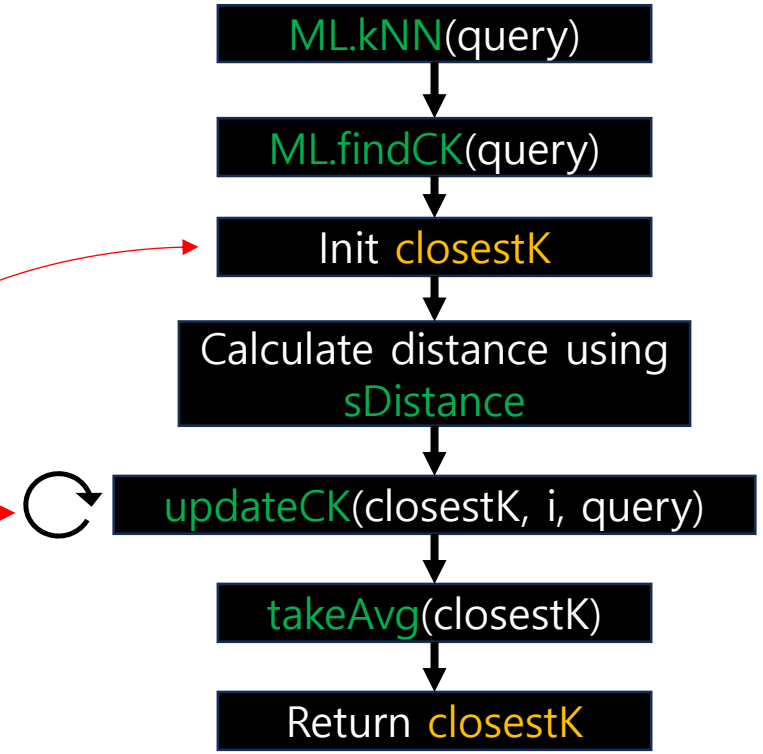
query에 대해 가장 가까운 K개의 이웃 찾는 함수
전체 training data N개 중에 K개를 선택(closestK)
나머지 N-K개를 확인하며 더 가까우면 closestK를 update
-sDistance, updateCK 메소드 사용

ML.sDistance(self, dataA, dataB)

두 데이터 간의 거리 계산

ML.takeAvg(self, closestK)

closestK의 평균 계산



K-NN 구현 – kNN (20분)

```
        def kNN(self, query):
                closestK = self.findCK(query)
                predict = self.takeAvg(closestK)
                return predict
```

```
query: array([-17.89, -21.4 , -1.68])
```

```
closestK (where K=5)
array([[346, 35], [ 54, 23], [136, 44], [351, 58], [203, 54]])
```

```
self._trainDX
array([[-1.132e+01, -1.794e+01, 2.490e+00],
[-9.130e+00, 4.080e+00, 1.713e+01],
[-2.475e+01, -1.560e+01, -2.706e+01],
 ...,
[ 3.640e+00, 2.610e+00, -2.255e+01],
[-2.570e+00, -2.131e+01, 2.000e-02],
[-2.665e+01, -2.869e+01, 1.825e+01]])
```

query는 [x,y,z] array로 주어짐

findCK가 반환해야 하는 값은
List of [index, distance]

Index? Self.trainDX에서 query와
가까운 원소를 지시하는 index

K-NN 구현 - findCK

```
def findCK(self, query):
    # implementation
    return closestK
```

```
def updateCK(self, closestK, i, query):
    # implementation
    return distance
```

```
def sDistance(self, dataA, dataB):
    # implementation
    return distance
```

1. **closestK 변수 초기화하기 (np.array, shape=(self._k, 2))**
   K개의 index, distance를 담을 변수임
   index는 0, 1, 2, … K 로 초기화

2. **closestK의 각 index에 대해 distance 계산해주기 (Euclidean distance)**
   (x,y,z), (x,y,z)를 받았을 때 거리를 반환해주는 **self.sDistance 함수 구현**

3. **closestK update 해주기**
   Train data의 K+1부터 끝까지 모든 점을 closestK와 비교
   현재 closest K와의 거리들 보다 더 가까운 점을 찾았으면
   가장 멀리 있는 closest K를 새로운 점으로 바꿔주기
   self.UpdateCK(closestK, I, query) 함수 구현 (i번째 train_dx와 closestK 비교하여 업데이트하는 함수)

K-NN 구현 - findCK

```
    def findCK(self, query):
        m = np.size(self._trainDy) # Number of training data
        k = self._k
        closestK = np.arange(2 * k).reshape(k, 2) # Prepare a k x 2 matrix
        for i in range(k):
            closestK[i, 0] = i
            closestK[i, 1] = self.sDistance(self._trainDX[i], query)
        for i in range(k, m):
            self.updateCK(closestK, i, query)
        return closestK
```

K-NN 구현 - sDistance

```
    def sDistance(self, dataA, dataB): # Return the squared distance
        dim = np.size(dataA)  # data dimension
        sumOfSquares = 0
        for i in range(dim):
            sumOfSquares += (dataA[i] - dataB[i]) ** 2
        return sumOfSquares
```

K-NN 구현 - updateCK

```
    def updateCK(self, closestK, i, query): # i replaces the worst if better
        d = self.sDistance(self._trainDX[i], query)

        j = np.argmax(closestK[:, 1])
        if closestK[j, 1] > d:
            closestK[j, 0] = i
            closestK[j, 1] = d
```

K-NN 구현 - takeAvg

```
def takeAvg(self, closestK):
    return total/K
```

ClosestK의 평균 값 구하는 함수 구현
*distance 평균 아님! trainDy에 대한 평균

K-NN 구현 - takeAvg

```
def takeAvg(self, closestK):
    k = self._k
    total = 0
    for i in range(k):
        j = closestK[i, 0]
        total += self._trainDy[j]
    return total / k
```

K-NN로 문제 풀어보기 (10분)

```
Enter the file name of training data: lin_train.txt
Enter the file name of test data: lin_test.txt

Which learning algorithm do you want to use?
 1. Linear Regression
 2. k-NN
Enter the number: 2
Enter the value for k: 5

RMSE:  0.91
```

```
Enter the file name of training data: lin_train.txt
Enter the file name of test data: lin_test.txt

Which learning algorithm do you want to use?
 1. Linear Regression
 2. k-NN
Enter the number: 2
Enter the value for k: 1

RMSE:  1.12
```

Linear, non-linear 문제에 대해 k 값에 따른 차이 비교

각 문제에 대한 Best K 값 찾기

과제: K-NN, Linear Regression 비교 실험

K-NN으로 찾은 최적의 K 값에서의 Linear, Non-Linear 결과와
Linear Regression으로 찾은 Linear, Non-Linear 결과 비교해보기

	K-NN (K=x1)	Linear Regression
Linear Problem	**?**	**?**

	K-NN (K=x2)	Linear Regression
Non-Linear Problem	**?**	**?**

과제

두 알고리즘 비교해보고 결과 확인

HW10 제출 필요 없음