AI 프로그래밍
HW10 실습

2021.12.09

코딩 환경 준비하세요

contents

- 1. HW10 전체 목표
- 2. today 실습 내용
- 3. 제출사항
- 4. 출석체크

1. HW10 전체 목표

- 1) KNN 구현
- 2) rearrange for “class ML”
- 3) experiments results

2. Today 실습 내용

main()	class ML:	class LinearRegression(ML):	class KNN(ML):
(skeleton) buildModel()	__init__(self) setData(self, dtype, fileName) createMatrices(self, fileName)		
	testModel(self) ← (skeleton) testKNN, testLR	__init__(self) buildModel(self) ← (skeleton) linearRegression runModel(self, data) ← (skeleton) LR	__init__(self) buildModel(self) ← input k runModel(self, query) ← (skeleton) kNN
			findCK(self, query) sDistance(self, data, dataB) updateCK(self, closestK, i, query)
			takeAvg(self, closestK)
	report(self) ← (skeleton) calcRMSE		

def kNN(self, query)

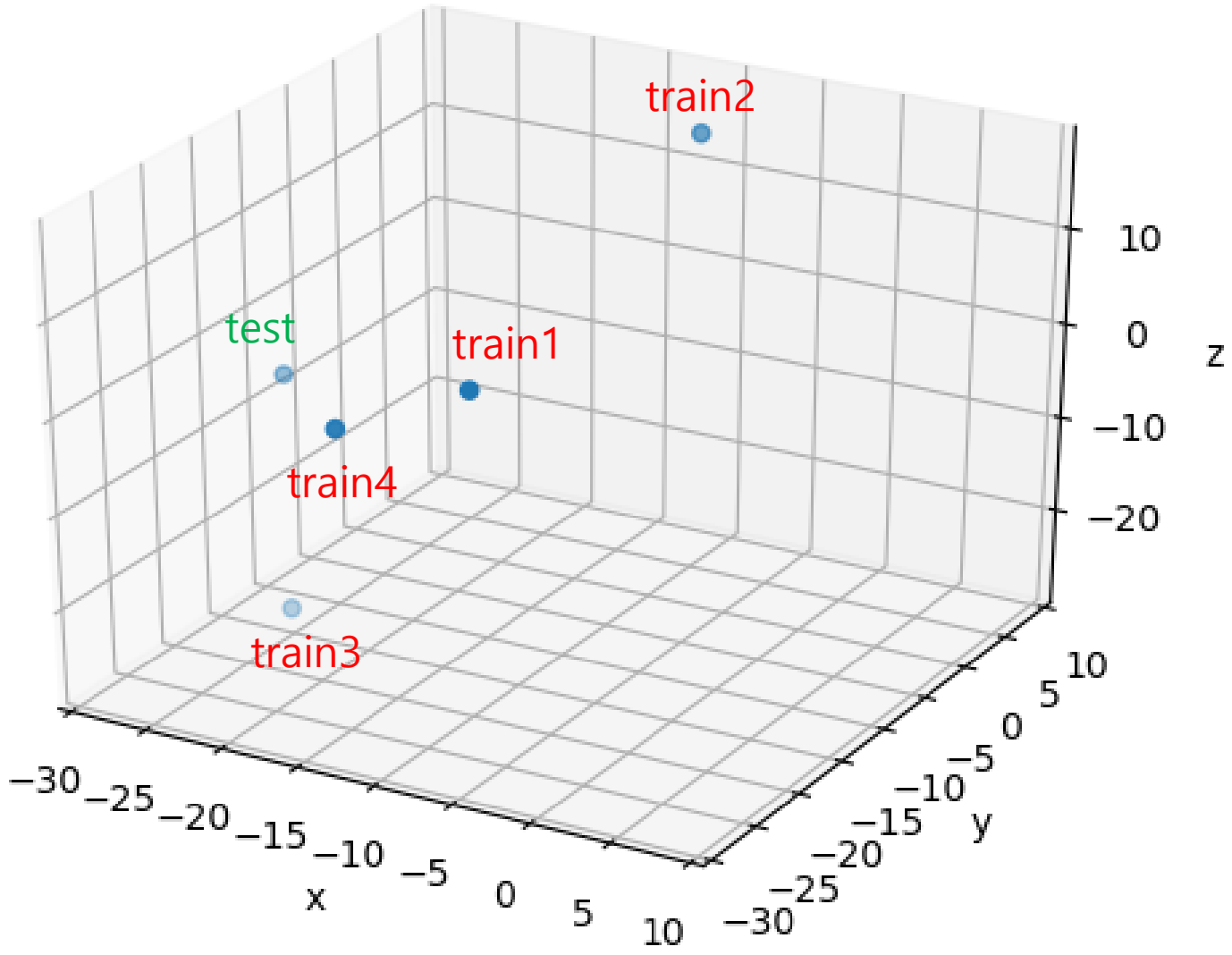
- findCK

query (= testDx[i]) 와
가장 가까이 있는 (이웃한)
K개의 trainDx[i] 찾기
(→closestK)

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

- takeAvg

찾은 closestK의 값 (거리)
평균 구하기
(→ predict)


train2
test
train1
train4
train3
10
0
−10
−20
z
−30
−25
−20
−15
−10
−5
0
5
10
x
10
5
0
−5
−10
−15
−20
−25
−30
y

def sDistance(self, data, dataB)

- trainDx[i]와 query(=testDx[i])의 거리 구하는 함수

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

def findCK(self, query)

- 처음 K개만큼의 trainDx[i]와 query(=testDx[i])의 거리 구하고,
- updateCK에서 k번째 다음것과의 거리 구해서 작으면 업데이트

```
def findCK(self, query):
    m = np.size(self._trainDy) # Number of training data
    k = self._k
    closestK = np.arange(2 * k).reshape(k, 2)
    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
```

def updateCK(self, closestK, i, query)

- k번째 다음것과의 거리 구해서
- 거리가 현재 closestK의 거리보다 작으면 업데이트

```
def updateCK(self, closestK, i, query): # i replaces the worst if better
    d = self.sDistance(self._trainDX[i], query)
    for j in range(len(closestK)):
        if closestK[j, 1] > d:
            closestK[j, 0] = i
            closestK[j, 1] = d
            break
```

def takeAvg(self, closestK)

- findCK에서 찾은

query (=testDx[i]) 와
가장 가까이 있는 (이웃한)
K개의 trainDx[i]와의
거리 평균값 구하기

```
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
```

3. 제출사항

- 보고서에 2개 문제(linear, non-linear) X 2개 알고리즘(LR, kNN) 비교내용 작성
- 결과 터미널 스크린샷 첨부
- 각각의 RMSE값 기재
- (kNN의 경우 최고의 k를 찾기 위해 k값을 바꿔가면서 실험한 내용 작성)
- discussion