지능시스템

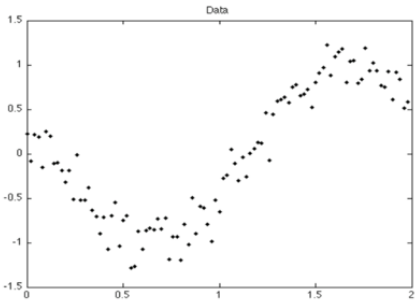
과제3

2019305059

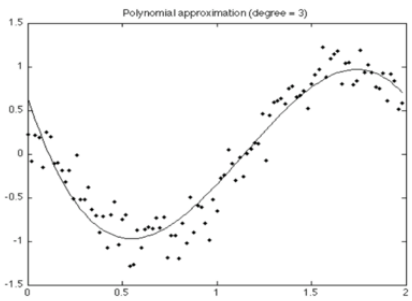
이현수

숙제 3 (경사하강법을 이용한 근사함수)

- 다음 그림과 같이 주어진 데이터에 대해 오차를 최소화하는 근사함수(3차 다항함수, 파라미터 (a, b, c, d))를 경사하강법으로 구하는 예제를 프로그램을 작성하여 실습하고 결과를 정리하여(학습의 중간 과정 포함, 예를 들어, 10 epoch학습 결과, 100 epoch학습 결과 등등) 보고서 형태로 제출한다.
- 데이터 쌍 (x_i, y_i)은 300개를 다음과 같이 생성하여 사용한다.
- $[0, 3]$ 구간의 x_i를 랜덤하게 발생시켜 $y = x^3 - 4.5x^2 + 6x + 2$의 함수에 대입하여 y를 구하고 여기에 $[-0.5, 0.5]$ 사이의 값을 랜덤하게 생성하여 더한 값을 y_i로 한다. (4월 16일까지 Google Classroom 제출)



(x_i, y_i) for $i = 1, 2, \cdots, N$



$y = ax^3 + bx^2 + cx + d$

매개변수 학습의 개념 – 예제 구현 코드

```
def linear_regression (X, y, m_current=0, b_current=0, epochs=1000,
learning_rate=0.0001):
        N = float(len(y))
        for i in range(epochs):
                y_current = (m_current * X) + b_current
                cost = sum([data**2 for data in (y-y_current)]) / (2*N)
                m_gradient = -(1/N) * sum(X * (y - y_current))
                b_gradient = -(1/N) * sum(y - y_current)
                m_current = m_current - (learning_rate * m_gradient)
                b_current = b_current - (learning_rate * b_gradient)
        return m_current, b_current, cost
```

1.데이터 쌍 (x, y) 300개 만들기

```python
import random

def makePairData(numOfData):
    x=[]
    y=[]

    for i in range(numOfData):
        temp_x=random.uniform(0,3)
        temp_y=(temp_x*temp_x*temp_x)-(4.5*temp_x*temp_x)+(6*temp_x) + 2 + random.uniform(-0.5,0.5)
        x.append(temp_x)
        y.append(temp_y)

    temp = [[a,b] for a, b in zip(x,y)]
    temp.sort(key=lambda x:x[0])
    x=[a[0] for a in temp]
    y=[b[1] for b in temp]

    return x,y

```

- 데이터 쌍 (x, y)은 300개를 생성한다.

- [0, 3] 구간의 x를 랜덤하게 발생시켜 y = x^3 – 4.5x^2 + 6x +2x의 함수에 대입하여 y를 구하고 여기에 [-0.5, 0.5] 사이의 값을 랜덤하게 생성하여 더한 값을 y로 한다.

- 이때 랜덤한 값을 생성할 때 파이썬 모듈 random의 uniform메소드를 통해 생성한다.

- x를 기준으로 sort 메소드를 통해 오름차순한 x, y를 반환한다.

2.경사하강법을 이용한 근사함수

```python
def linear_regression(X, Y, a_current, b_current, c_current, d_current, epochs, learning_rate):
    N=len(Y)
    for i in range(epochs):
        cost=0
        a_gradient=0
        b_gradient=0
        c_gradient=0
        d_gradient=0
        y_current=[]
        for j in range(N):
            y_current.append((a_current*X[j]**3)+(b_current*X[j]**2)+(c_current*X[j])+(d_current))
            a_gradient += (X[j]**3) * (Y[j] - y_current[j])
            b_gradient += (X[j]**2) * (Y[j] - y_current[j])
            c_gradient += X[j] * (Y[j] - y_current[j])
            d_gradient += Y[j] - y_current[j]
            cost += (Y[j] - y_current[j]) ** 2

        cost/=(2*float(N))
        a_gradient = -(1 / float(N)) * a_gradient
        b_gradient = -(1 / float(N)) * b_gradient
        c_gradient = -(1 / float(N)) * c_gradient
        d_gradient = -(1 / float(N)) * d_gradient

        a_current = a_current - (learning_rate * a_gradient)
        b_current = b_current - (learning_rate * b_gradient)
        c_current = c_current - (learning_rate * c_gradient)
        d_current = d_current - (learning_rate * d_gradient)
        if i%100==0:
            print('Epoch:' + str(i) + '==============================================\
==========================================================')
            print(f'y = {a_current}x^3 + {b_current}x^2 + {c_current}x + {d_current}')
            print('cost = ' + str(cost))

    print('Epoch:' + '1000' + '==============================================\
==========================================================')
    print(f'y = {a_current}x^3 + {b_current}x^2 + {c_current}x + {d_current}')
    print('cost = ' + str(cost))

    return a_current, b_current, c_current, d_current, cost
```

- 예제구현 코드를 기반으로 수정해서 함수를 만든다.

- 100에포크 단위로 중간 결과를 출력한다.

3. 실행

```
59
60
61      numOfData = 300
62      x,y=makePairData(numOfData)
63
64      for i in range(len(x)):
65          print(f'{i+1}: {x[i]}, {y[i]}')
66
67      a,b,c,d,cost = linear_regression(x,y,0,0,0,0,1000,0.0001)
68
69      print()
70      print('result:')
71      print(f'a = {a}, b = {b}, c = {c}, d = {d}')
```

- 앞서 만든 makePairData 함수와 linear_regression 함수를 이용해 3차 다항함수 파라미터 a, b, c, d를 구한다.

- 1000에포크로 learning_rate=0.0001으로 실행했다.

4. 결과

```
1: 0.004570026689213136, 1.9031382272849509
2: 0.008482980622597003, 2.0051241619675757
3: 0.018068500372705132, 2.032683694486323
4: 0.01832597731545771, 1.627034405018789
5: 0.028476026038341318, 2.608055518526055
6: 0.05825602667766283, 1.973189474352625
7: 0.08155240418239151, 2.574118440727959
8: 0.08292101594642565, 2.017790592977618
9: 0.1002934079808917, 2.620663053487868
10: 0.1043024165434775, 2.400131642337951
```

```
291: 2.8479651783613127, 6.0714464995257895
292: 2.860315441296865, 5.305146116380241
293: 2.868960702910292, 5.6552346496065455
294: 2.908203210890678, 5.629541431022604
295: 2.923133427943918, 5.704509358127882
296: 2.9311340596015185, 6.55212534481819
297: 2.944255046283368, 6.2488914719130175
298: 2.9479615681397906, 6.5014711011714
299: 2.9769610305138414, 6.713822939433311
300: 2.978747263448665, 6.281363033563718
```

1. makePairData 함수를 통해 만들어진 300개의 데이터이다.

```
Epoch:0=========================================================================================================
y = 0.0031342596467077454x^3 + 0.0013635105665898099x^2 + 0.0006646256193404852x + 0.0004197945378128337
cost = 9.110054543831744
Epoch:100=======================================================================================================
y = 0.18180140865486366x^3 + 0.08452677767005917x^2 + 0.04549448185429292x + 0.03314086199906958
cost = 4.357324580877987
Epoch:200=======================================================================================================
y = 0.2306990593105357x^3 + 0.11638512803855426x^2 + 0.06939397015536093x + 0.056838154198335974
cost = 3.851995538234745
Epoch:300=======================================================================================================
y = 0.2403588613695926x^3 + 0.13261443926653219x^2 + 0.08683216522635112x + 0.07766138408901596
cost = 3.737822934799225
Epoch:400=======================================================================================================
y = 0.2381989437121478x^3 + 0.14402073785224892x^2 + 0.10219264601304749x + 0.09747467231905961
cost = 3.6610488828726337
Epoch:500=======================================================================================================
y = 0.23252256298638627x^3 + 0.15387782918396362x^2 + 0.11680443111802027x + 0.11684416526122043
cost = 3.589189752124558
Epoch:600=======================================================================================================
y = 0.22584375718789113x^3 + 0.16317826014677211x^2 + 0.1310714037095881x + 0.1359428803544847
cost = 3.5192322957378397
Epoch:700=======================================================================================================
y = 0.21892322052995658x^3 + 0.1722236524003247x^2 + 0.14511724012017524x + 0.1548247523528764
cost = 3.4508672661596558
Epoch:800=======================================================================================================
y = 0.21199062886241357x^3 + 0.181106358907401x^2 + 0.15898073636780743x + 0.17350763256238844
cost = 3.3840341910315908
Epoch:900=======================================================================================================
y = 0.20511483924796195x^3 + 0.1898553410028774x^2 + 0.1726749659123727x + 0.19199842660953736
cost = 3.318696145352335
Epoch:1000======================================================================================================
y = 0.19838359456905794x^3 + 0.19839469816843158x^2 + 0.18607069577218396x + 0.21011860956461698
cost = 3.2554507236325696
```

2. 100에포크 단위로 결과를 출력했다.

```
result:
a = 0.19838359456905794, b = 0.19839469816843158, c = 0.18607069577218396, d = 0.21011860956461698
```

3. 최종적으로 만들어진 a, b, c, d를 출력했다.