인공지능시스템 HW 2

제출기한 : 23년 4월 28일 (금) 23:59분

모든 문제에 대해 작성한 프로그램 코드 및 실행 결과 캡처본을 포함하여 word로 작성
작성된 Programming 코드 (.py 또는 ipynb) 파일 함께 제출
word파일과 코드 파일을 하나의 .zip파일로 압축하여 제출
제출 파일명 형식 : 홍길동_211234_HW02.zip (미 준수 시 감점)
본인이 작성한 모든 프로그래밍 코드에는 주석을 상세히 달아 자세히 설명 (누락 시 감점)
COPY 적발 시 해당 과제는 0점 처리
제출 딜레이 : 1주 당 25%씩 점수 감점

[참고 1]

: Python 에서는 다음과 같이 2차 함수 방정식을 symbolic으로 연산/미분 할 수 있다.
*sympy : 실제 수식처럼 기호 연산이 가능하도록 제공되는 python 모듈

```
import sympy as sym
x = sym.Symbol('x') # x라는 심볼 객체 생성 (심볼 객체 일반적인 변수와는 다른 형태의 데이터형임)
y = x**2+2*x+1 # x^n을 심볼릭으로 표현하기 위해서는 x**n 으로 표현 (예. x**2는 x^2를 나타냄)
y  # y는 x에 대한 심볼릭으로 표현된 함수. 출력 시 아래와 같음
Out[81]: x^2 + 2x + 1

y_ = sym.diff(y) # sym.diff 함수를 이용하여 y의 수식을 미분함
y_ # 미분한 결과를 출력하면 아래 와 같음
Out[82]: 2x + 2

y_.subs(x,2) # subs 함수를 이용하여, 해당 수식의 특정 변수에 숫자를 넣어줄 수 있다. (예시. 2x+2 의 x에 숫자 2를 넣어줌)
Out[118]: 6

result = float(y_.subs(x,2)) # 심볼릭 연산의 결과값은 일반적인 실수 형태의 float 데이터가 아니며, 변환은 왼쪽과 같이 한다.
print(result)
```

[참고 2]

: numpy 모듈을 활용하면 행렬의 역행렬을 다음과 같이 구할 수 있으며, 연립 방정식의 해 x도 구할 수 있다.

$\begin{bmatrix}133275 & 815\\ 815 & 5\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}46875\\ 285\end{bmatrix}$, A=$\begin{bmatrix}133275 & 815\\ 815 & 5\end{bmatrix}$, x=$\begin{bmatrix}x\\ y\end{bmatrix}$, b=$\begin{bmatrix}46875\\ 285\end{bmatrix}$

행렬의 연산에 의해 $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

```
import numpy as np
A = [[133275, 815], [815, 5]]
b = [46875, 285]
inv_A = np.linalg.inv(A) # 2x2 행렬의 역행렬 (A^-1) 계산
result = np.dot(inv_A,b) # A^-1 b dot product 계산
print('x는 %.4f, y는 %.4f 입니다.' % (result[0], result[1]))
```

실행 예)

```
x는 0.9767, y는 -102.2093 입니다.
```

[참고 3]

```
f = open("linear_regression.txt",'r') # linear_regression.txt 파일을 읽기 전용으로 open
lines = f.readlines() # 모든 라인의 데이터를 load

for line in lines:
   line_data = line.split(' ') # split 명령어 안에 있는 string 데이터로 데이터를 나누어 리스트형태로 변환하는 함수
                    # (‘ ’)은 공백을 기준으로 데이터를 나눔
   print(line_data)
f.close()
```

실행 예)

```
['A', '2.5', '11\n']
['B', '3.2', '14\n']
['C', '4.1', '21\n']
['D', '5.0', '27\n']
['E', '6.3', '33\n']
['F', '7.0', '36\n']
['G', '8.2', '43\n']
['H', '9.1', '48\n']
['I', '10.3', '52\n']
['J', '11.5', '57\n']
```

[참고 4]

```
import pandas as pd
import numpy as np

pd_data = pd.read_csv('train_data.csv') # train_data.csv 파일을 pandas DataFrame 으로 읽어오기

features = pd_data.iloc[:, :-1] # 4 가지 특징에 대해 데이터 선택
target = pd_data.iloc[:, -1:] # 클래스 부분만 선택
```

실행 예) features

	like_human	like_walk	like_box	like_sleep
0	1	1	0	0
1	1	1	0	0
2	1	1	0	0
3	1	1	0	0
4	1	1	0	0
...	...	...	...	...
95	0	0	0	1
96	0	0	1	0
97	0	0	1	0
98	1	0	1	0
99	1	0	1	0

100 rows × 4 columns

실행 예) target

	class
0	Dog
1	Dog
2	Dog
3	Dog
4	Dog
...	...
95	Cat
96	Cat
97	Cat
98	Cat
99	Cat

100 rows × 1 columns

```
# column 'like_human'에 존재하는 값, 각 값들에 대한 개수 반환
value, counts = np.unique(pd_data['like_human'], return_counts=True)
print(value, counts)
```

실행 예) ‘like_human’ column 에 value 0 에 대해 48 개, value 1 에 대해 52 개 존재

```
[0 1] [48 52]
```

1. Numerical differential (미분의 근사치 구하기)

1-1) 위의 [참고 1] 코드에서는 심볼 객체를 활용하여 $y = x^2 + 2x + 1$ 를 정의하였고, 그 도함수 $y' = 2x + 2$ 를 구했고, $y'(2) = 6$ 임을 계산했다. 심볼 객체를 사용하지 않고, $y'(2)$를 근사적으로 구해서 반환하는 함수 numerical_diff(x) 를 작성하라. [7.5점]

(hint. 함수의 입력 x 지점(p1)과 아주 가까운 위치의 한점을 p2를 임의로 설정하고 그 두 점 사이의 기울기를 구한다.)

```
# numerical differential
def numerical_diff(x):
    p1 = x*x+2*x+1
    ### 함수 내부 코드 작성

    return diff_result

print("x^2+2x+1 함수의 x=2지점에서의 미분값은", numerical_diff(2), "입니다.")
```

실행 예)

`x^2+2x+1` 함수의 `x=2`지점에서의 미분값은 `6.000000087880153` 입니다.

1-2) 문제 1-1)에서 구한 numerical differential 값과, [참고 1]에서 구한 symbolic differential 값 간의 오차는 얼마나 되는가? 이 오차를 줄이기 위해서는 numerical differential 함수를 어떻게 설계 해야 하는가? numerical differential 로 얻은 미분 근사치가 유의미한 값이라고 판단 되는가? [7.5 점]

2. 선형 회귀 모델 (linear regression)

2-1) Linear_regression.txt 파일을 파일 입출력 객체를 활용하여 읽고 각각 list 에 데이터에 저장하시오. [2.5 점]

[참고 3]의 코드를 참고하세요

```
A 2.5 11
B 3.2 14
C 4.1 21
D 5.0 27
E 6.3 33
F 7.0 36
G 8.2 43
H 9.1 48
I 10.3 52
J 11.5 57
```

* linear_regression.txt 파일

1열 : 인덱스
2열 : 작업 시간 (Hours Worked)
3열 : 인형 수 (Number of Dolls)

예) 1행 데이터
A, 2.5 hours, 11 dolls

2-2) 최소제곱법(Least square method)를 이용하여, 문제 2-1)에서 읽은 데이터에 대한 linear regression(선형회귀 모델 학습)을 수행하라 [15 점]

※ numpy 패키지 활용 가능 (sklearn 과 같은 머신러닝 패키지 활용 불가능)

[참고 1,2]의 내용을 참고하세요.

Linear regression 을 수행한 후 파라미터들 값을 꼭 출력할 것

2-3) 구해진 선형 모델 및 학습 데이터를 matplotlib 를 이용하여 시각화하시오. [2.5 점] (첨부파일 '2 차원그래프.ipynb 참고')

2-4) 학습된 선형 회귀 모델에 의해 7.3 시간 작업하였을 때 완성한 인형 수는 몇 개라 예측할 수 있는가? (코드로 작성) [5 점]

hint) 예측한 개수가 정수가 아닐 수도 있음

실행 예)

학습 모델에 의해 예측한 `7.3` 시간 작업하였을 때 완성된 인형은 `00` 개입니다.

3. 선형 분류 모델 (linear classification)

3-1) Linear_classification.txt 파일을 파일 입출력 객체를 활용하여 읽고 각각 list 에 데이터에 저장하시오. [2.5 점]

[참고 3]의 코드를 참조하세요

A 35 27 1 B 81 8 -1 C 16 38 1 D 28 42 1 E 72 20 -1 F 64 43 -1 G 29 13 1 H 41 22 -1 I 32 51 -1 J 15 36 1 * linear_classification.txt 파일	1열 : 각 실험 인덱스 2열 : 용액S 용량 (ml) 3열 : 용액T 용량 (ml) 4열 : label (+1인 경우 실험 성공 / -1인 경우 실험 실패) 예) 1행 데이터 실험 A , 용액S 35ml, 용액T 27ml, 실험 성공

3-2) 최소제곱법(Least square method)를 이용하여, 문제 3-1)에서 읽은 데이터에 대한 linear classification(선형분류 모델 학습)을 수행하라[15 점]

※ numpy 패키지 활용 가능 (sklearn 과 같은 머신러닝 패키지 활용 불가능)

[참고 1,2]의 내용을 참고하세요

Linear classification 을 수행한 후 파라미터들 값을 꼭 출력할 것

3-3) 구해진 선형 모델 및 학습 데이터를 matplotlib 를 이용하여 시각화하시오. [2.5 점] (첨부파일 '3 차원그래프.ipynb 참고')

3-4) 학습된 선형 분류 모델에 의해 용액 S 47ml, 용액 T 29ml 인 실험은 성공인가? 실패인가? (코드로 작성) [5 점]

hint) 모델로 예측된 값이 0 보다 클 경우 실험 성공, 0 보다 작을 경우 실험 실패로 판단

실행 예)

입력한 실험은 학습 모델에 의해 실험 성공으로 판단됩니다.

4. Decision Tree (결정 트리)

4-1) train_data.csv 파일의 학습데이터 100 개를 이용하여 decision tree (결정 트리)를 학습하시오. [25 점]

※ numpy, pandas 라이브러리활용 가능 (기타 머신러닝 패키지 ex. skelarn.DecisionTreeClassifier 활용 불가능)

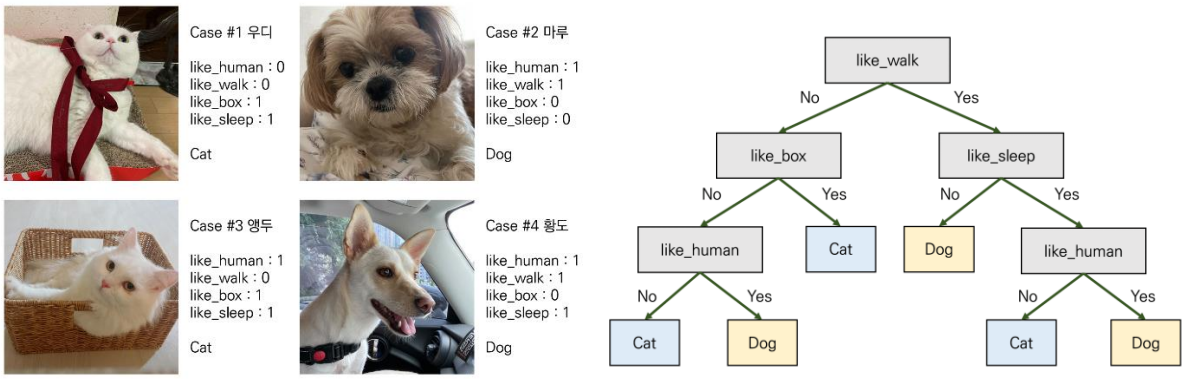
[참고 4]의 코드를 참고하세요

-Entropy 계산은 수업 시간 진행한 방식대로 할 것.

-결정트리는 이진 트리로 작성할것.

-코드 상 출력한 decision tree 를 토대로 아래 예시의 오른쪽과 같이 decision tree 를 그려 코드 실행 결과와 함께 첨부할 것.

Ex) 아래 데이터, decision tree 는 이해를 돕기 위한 단순 예시입니다.



4-2) test_data.csv 파일의 테스트 데이터 10 개에 대한 테스트 결과를 출력하시오. [10 점]

실행 예)

```
Test #0 [1, 1, 0, 0, 'Dog'] -> Dog
Test #1 [0, 0, 1, 1, 'Cat'] -> Cat
Test #2 [1, 1, 0, 0, 'Dog'] -> Dog
…
Test #9 [1, 1, 1, 0, 'Dog'] -> Dog
```