

[Machine Learning]

[2021-1]

Homework 1

[Due Date] 2021.04.02

Student ID : 2016112158

Name : KimHeeSu

Professor : Juntae Kim



1. Write python codes to solve each of the following problem, and attach the result and description. (20 pts)

1-1. Python : Circle and Rectangle Class design (Week02-Quiz4)

Code	
<pre>class Circle: def __init__(self, radius): self.radius = radius def area(self): return self.radius ** 2 * 3.14 # make Rectangle class class Rectangle: def __init__(self, width, height): self.width = width self.height = height def area(self): return self.width * self.height</pre>	
Result(Captured images)	
	
Description	
<p>Circle 클래스의 생성자는 radius 를 파라미터로 받고, 그 값으로 self.radius(멤버)를 초기화한다. Circle 클래스의 area 메소드는 self.radius 로 원의 넓이를 계산하는 메소드이다.</p> <p>Rectangle 클래스의 생성자는 height(높이)와 (width)를 파라미터로 받고 그 값으로 self.height 와 self.width 를 초기화한다. area 메소드는 self.width 와 self.height 로 직사각형의 넓이를 계산하는 메소드이다.</p>	

1-2. Numpy : Matrix Dot Product

For $X = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$, $w = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}$,

Compute $y = \begin{pmatrix} \\ \\ \end{pmatrix}$ where

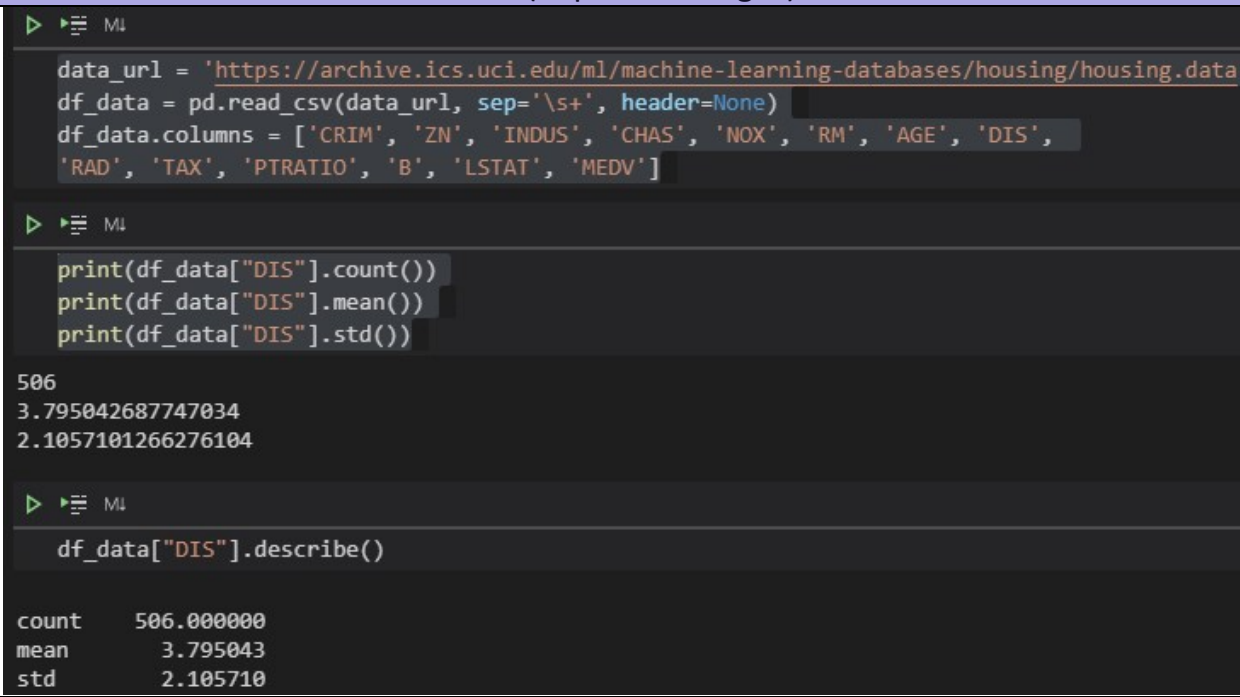
$$y_j = \sum_i (w_i \cdot x_{ij})$$

Use these functions:

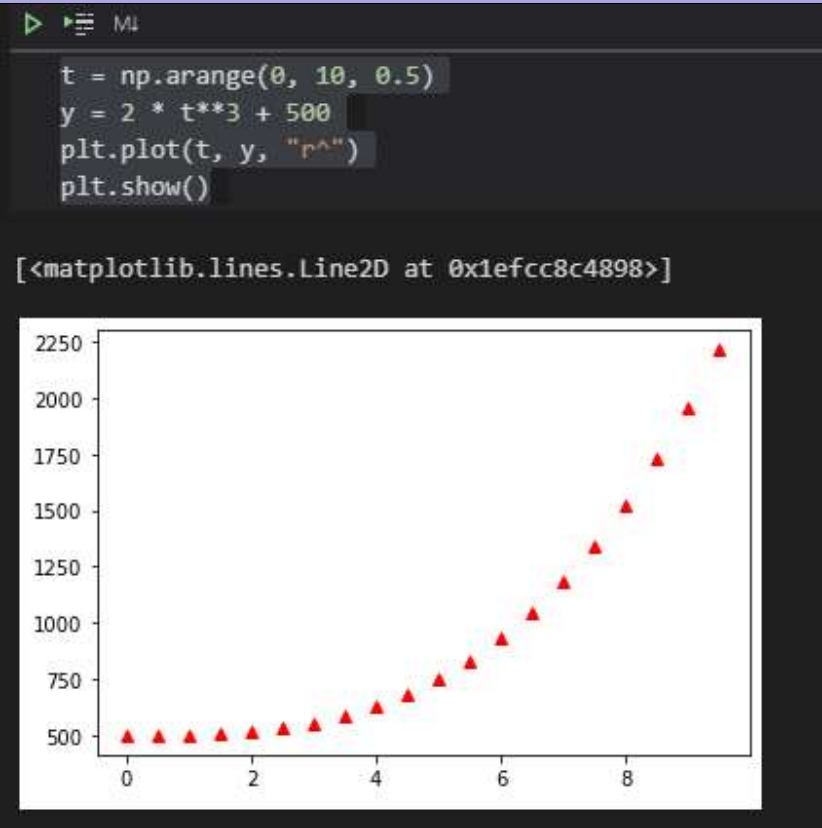
- `np.array()`, `np.arange()`, `np.dot()`
- `X.reshape()`, `X.T`

Code	
<pre>X = np.arange(1,13).reshape((3,4)) w = np.array([0.1,0.2,0.3]) y = X.T.dot(w) y</pre>	
Result(Captured images)	
Description	
<p>X 를 생성할 때 <code>np.arange</code> 와 <code>reshape</code> 를 사용해서 shape 가 (3,4)인 matrix 생성 W 는 <code>np.array</code> 를 이용해 생성한다. X.shape 는 (3,4)이고 w.shape 는 (3,)이다. Shape 가 (4,)인 y 를 주어진 식대로 계산하려면 matrix dot product 를 이용하면 된다. Dot product 를 하려면 행과 열을 같게 맞춰줘야 하므로 transpose 메소드 T 를 사용해 (4,3) x (3,) = (4,)이므로 y.shape 와 같다.</p>	

5-4. Pandas : From Boston Housing Price dataset, compute “DIS” column’s count, mean, std. (Week03-Quiz4)

Code
<pre>data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data' df_data = pd.read_csv(data_url, sep='\s+', header=None) df_data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV'] print(df_data["DIS"].count()) print(df_data["DIS"].mean()) print(df_data["DIS"].std())</pre>
Result(Captured images)
 <pre>data_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data' df_data = pd.read_csv(data_url, sep='\s+', header=None) df_data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV'] print(df_data["DIS"].count()) print(df_data["DIS"].mean()) print(df_data["DIS"].std()) 506 3.795042687747034 2.1057101266276104 df_data["DIS"].describe() count 506.000000 mean 3.795043 std 2.105710</pre>
Description
<p>Pandas series 는 column 내 인스턴스 개수를 계산하는 count, 인스턴스들의 평균을 계산하는 mean, 표준편차를 계산하는 std 메소드를 가지고 있다. Describe 메소드는 count, mean, std 뿐만 아니라 min,max, 4 분위수들까지 계산해준다</p>

1-4. Matplotlib : Plot $y = 2t^3 + 500$ for $t = [0, 0.5, 1.0, 1.5, \dots, 8.5, 9.0, 9.5]$ with red triangles.

Code	
<pre>t = np.arange(0, 10, 0.5) y = 2 * t**3 + 500 plt.plot(t, y, "r^") plt.show()</pre>	
Result(Captured images)	
 <p>The screenshot shows a Jupyter Notebook interface. The code cell contains the following Python code:</p> <pre>t = np.arange(0, 10, 0.5) y = 2 * t**3 + 500 plt.plot(t, y, "r^") plt.show()</pre> <p>Below the code, the output is displayed, showing the return value of the plot function: <code>[<matplotlib.lines.Line2D at 0x1efcc8c4898>]</code>. Below this, a line plot is shown with red triangles. The x-axis ranges from 0 to 10, and the y-axis ranges from 500 to 2250. The plot shows a cubic curve starting at (0, 500) and increasing rapidly.</p>	
Description	
<p>Np.arange 를 이용해서 0~9.5 데이터를 쉽게 생성할 수 있다. 이후, 주어진 식대로 y 를 t 로 표현한 후, <code>plt.plot</code> 의 첫번째 파라미터로 t, 두번째 파라미터로 y 를 준 후, 세번째 파라미터로 $r^$를 주면 알아서 차트가 그려진다.</p>	

2. Explain what Supervised Learning, Unsupervised Learning, and Reinforcement Learning are, and describe the differences. (10 pts)

Your Answer
<p>지도학습(supervised learning)은 정답이 있는 데이터를 활용해 데이터를 학습시키는 것이다. 입력값(X)가 주어지면 입력값에 대한 Label(Y)를 주어 학습시킨다.</p> <p>비지도학습(unsupervised learning)은 정답 레이블이 없는 데이터를 비슷한 특징끼리 군집화하여 새로운 데이터에 대한 결과를 예측하는 방법이다. 라벨링되어 있지 않은 데이터로부터 패턴이나 형태를 찾아야 한다.</p> <p>강화학습(Reinforcement learning)은 상과 벌이라는 reward 를 주며 상을 최대화하고 벌을 최소화하도록 학습한다. 환경이 있고 에이전트가 그 환경속에서 어떤 액션을 취하고 그 액션에 따른 보상을 얻게 되며 학습이 진행되며 그 보상을 최대화하는 방향으로 학습이 진행된다</p>

3. Describe the concept of “overfitting”, and explain how you can prevent overfitting in supervised learning. (20 pts)

Your Answer
<p>과대적합(overfitting)은 필요이상의 feature 를 발견하여 훈련데이터에선 높은 정확도를 보이지만 테스트 데이터나 새로운 데이터에 대해선 정확도가 낮게 나오는 경우를 말한다. 즉, 훈련데이터에 지나치게 알맞아 일반화가 되지 않는 경우를 뜻한다. 데이터에서 feature 를 필요이상으로 추출하게 되면 분산(variance, 예측값의 변동성)이 높아져 과대적합이 발생하는데, 분산을 줄이면 과대적합을 피할 수 있다. 분산을 줄이는 방법으론 파라미터 수가 적은 모델을 선택하거나, 훈련데이터에 있는 feature 수를 줄이거나, 모델에 regularization 을 가하여 단순화시킨다.</p>

4. Describe the differences between Gradient Descent and Stochastic Gradient Decent in detail and explain pros and cons (you can explain by using examples). (20 pts)

Your Answer
<p>경사하강법(Gradient Descent, GD)은 배치(단일 반복에서 기울기를 계산하는데 사용하는 샘플의 총 개수)를 전체 데이터 셋(batch_size==len(train_set))으로</p>

두고 그래디언트를 계산하여 그래디언트 값에 따라 **weight** 를 업데이트 하는 방법이다. 그에 반해, **SGD** 는 반복당 하나의 샘플(즉, 배치 크기가 1)에 대해서 그래디언트를 계산한 후 **weight** 를 업데이트하는 방법이다. **GD** 는 전체 데이터에 대해 그래디언트를 계산하므로 최적해로의 수렴이 안정적(smooth)하게 진행되는 장점도 있지만 한 스텝에 모든 훈련데이터를 사용하므로 학습이 오래 걸린다. 구글 데이터 세트 같이 엄청나게 많은 특성이 있는 데이터가 또 엄청 많다면 시간이 매우 많이 소요될 것이다. **SGD** 는 랜덤하게 추출한 샘플데이터 하나를 사용하므로 학습 중간과정에서 결과의 진폭이 크고 불안정하나, 속도가 매우 빠르다.

5. The seeds.csv dataset represents 7 geometric parameters of wheat kernels for 3 different varieties of wheat. Preprocess the dataset properly and output the cost function graph when you perform AdalineGD and AdalineSGD respectively (Specify the hyperparameter – η , epoch, etc.). (30 pts)

Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

class AdalineGD(object):
    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        self.eta = eta # learning rate
        self.n_iter = n_iter # number of iteration
        self.random_state = random_state

        # weight initialization
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])

    def fit(self, X, y):
```

```

self.cost_ = []

for l in range(self.n_iter):
    net_input = self.net_input(X)
    output = self.activation(net_input)

    #####
    # w = w + eta * (X.T dot errors)
    errors = y - output
    self.w_[1:] += self.eta * X.T.dot(errors)
    self.w_[0] += self.eta * errors.sum()

    # compute cost
    cost = (errors**2).sum() / 2.0
    self.cost_.append(cost)
    #####
    print(self.w_)

return self

def net_input(self, X):
    return np.dot(X, self.w_[1:]) + self.w_[0]

def activation(self, X):
    return X

def predict(self, X):
    return np.where(self.activation(self.net_input(X)) >= 0.0, 1, -1)

class AdalineSGD(object):

    def __init__(self, eta=0.01, n_iter=10, shuffle=True, random_state=None):
        self.eta = eta
        self.n_iter = n_iter
        self.w_initialized = False
        self.shuffle = shuffle
        self.random_state = random_state
        self._initialize_weights(X.shape[1])

```

```

def fit(self, X, y):
    self.cost_ = []
    for l in range(self.n_iter):
        if self.shuffle:
            X, y = self._shuffle(X, y)
        cost = []
        for xi, target in zip(X, y):
            cost.append(self._update_weights(xi, target))
        avg_cost = sum(cost) / len(y)
        self.cost_.append(avg_cost)
        print(self.w_)
    return self

def _shuffle(self, X, y):
    r = self.rgen.permutation(len(y))
    return X[r], y[r]

def _initialize_weights(self, m):
    self.rgen = np.random.RandomState(self.random_state)
    self.w_ = self.rgen.normal(loc=0.0, scale=0.01, size=1 + m)
    self.w_initialized = True

def _update_weights(self, xi, target):
    output = self.activation(self.net_input(xi))
    error = (target - output)
    self.w_[1:] += self.eta * xi.dot(error)
    self.w_[0] += self.eta * error
    cost = 0.5 * error**2
    return cost

def net_input(self, X):
    return np.dot(X, self.w_[1:]) + self.w_[0]

def activation(self, X):
    return X

def predict(self, X):

```

```

        return np.where(self.activation(self.net_input(X)) >= 0.0, 1, -1)

seed_df = pd.read_csv("seeds.csv")
seed_df.columns = ["1","2","3","4","5","6","7","8"]

X = seed_df.iloc[:,[0,1,2,3,4,5,6]]
Y = seed_df.iloc[:,[-1]]

Y = np.array(Y).reshape(len(Y))

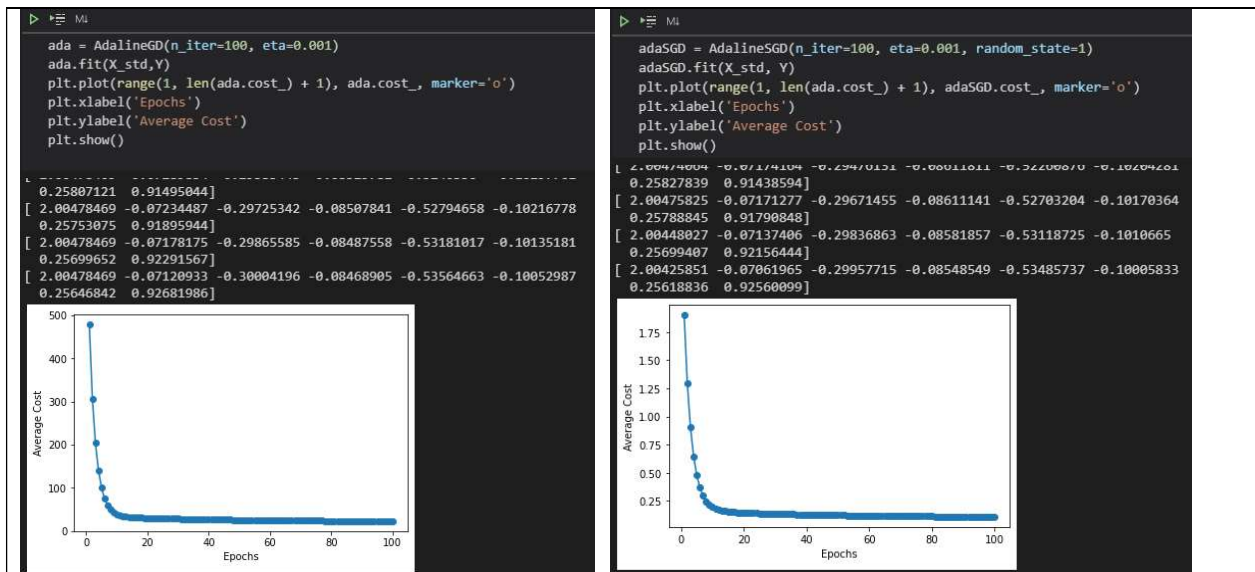
X_std = np.copy(X)
for l in range(7):
    X_std[:,l] = (X_std[:,l] - X_std[:,l].mean()) / X_std[:,l].std()

ada = AdalineGD(n_iter=100, eta=0.0001)
ada.fit(X_std,Y)
plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Average Cost')
plt.show()

adaSGD = AdalineSGD(n_iter=100, eta=0.01, random_state=1)
adaSGD.fit(X_std, Y)
plt.plot(range(1, len(adaSGD.cost_) + 1), adaSGD.cost_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Average Cost')
plt.show()

```

Result(Captured images)



Description

seed_df = pd.read_csv("seeds.csv")로 데이터프레임을 생성한다. seed_df에는 현재 column 이름이 없으므로 columns을 임의로 설정해준다. 데이터를 잘 살펴보면 column 8이 label에 해당하고 column 1~7은 feature에 해당한다. 이제 seed_df로 X, Y를 생성한다. 이때 reshape로 Y의 shape를 1차원으로 만들어줘야한다. 다음으로 X에 해당하는 각 column들의 scale이 모두 다르므로 정규화(평균 빼고 표준편차로 나누기)해준다. 정규화는 각 column에 대해 for문을 사용해도 좋지만 sklearn.preprocessing의 StandardScaler를 사용할 수도 있다. 이제 X, Y 데이터셋이 준비되었으므로 AdalineGD, AdalineSGD 모델 객체를 생성하고 훈련을 실시한 후, 시각화한다. 각 모델의 모두 learning rate는 0.001, epochs는 100번 반복했다.

Note

1. Submit the file to e-class as pdf
2. Specify your pdf file name as "hw1_<StudentID>_<Name>.pdf"
Ex) hw1 2000123456 홍길동.pdf