[Machine Learning]

[2022-1]

Homework 1

[Due Date] 2022.03.30

Student ID : 2017112138
Name : Yeojoon Jung
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: Design a Student class in Python. It will need to have the attributes **name**, **email**, **math_score**, **science_score**, and **english_score**. You will also need to include methods **average()** to calculate the average grade of a student and **email()** to print the email of the student.

Code

```python
class Student:
    def __init__(self,name,email,math_score,science_score,english_score):
        self.name = name
        self.email = email
        self.math_score = math_score
        self.science_score = science_score
        self.english_score = english_score

    def average(self):
        return (self.math_score + self.science_score + self.english_score)/3
    def email(self):
        print(self.email)

Std = Student('Yeojoon Jung','pucoy332@naver.com',100,90,80)
print("Average: ",Std.average())
Std.email
```

Result(Captured images)

```
Average:  90.0

'pucoy332@naver.com'
```

Description

set average() method to calculate average of math, science, English score and return average score.
email() method use print() function to print email of the student.
Creating objects of Student with name, email, math score, science score, English score.

1-2. Numpy: Matrix Dot Product

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

Compute $\mathrm{y} = \begin{pmatrix} \\ \\ \end{pmatrix}$ 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>import numpy as np
 	
X = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
w = np.arange(0.1,0.31,0.1)
y = np.dot(X.T,w)
y = y.reshape((4,1))
print(y)</pre>

Result(Captured images)

<pre>[[3.8]
 [4.4]
 [5.]
 [5.6]]</pre>

Description
Use np.array() to set X and use np.arange(start, stop, difference) to set w. Use np.dot() to calculate dot product. X.T is transpose of X. Original y`s shape is (1,4). Use y.reshape((4,1)) to reshape y to (4,1).

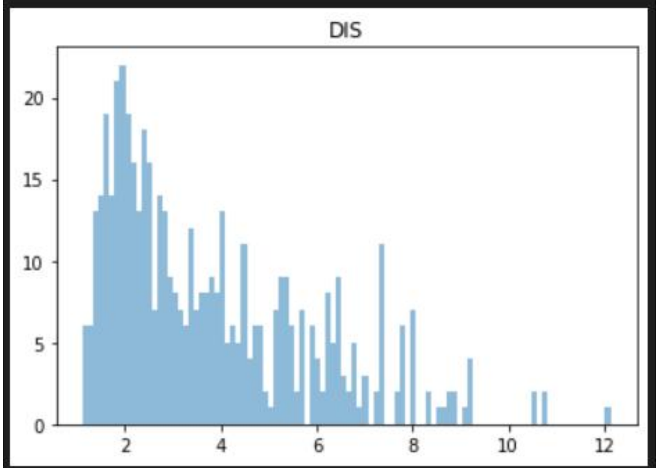
1-3. Pandas: From Boston Housing Price dataset, compute “DIS” column’s count, mean, std and model the distribution of “DIS” using a Histogram.

Code

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('./boston.csv')
print("DIS column`s count: ",df['DIS'].count())
print("DIS column`s mean: ",df['DIS'].mean())
print("DIS column`s std: ",df['DIS'].std())

plt.hist(df['DIS'], alpha=0.5,bins=100)
plt.title('DIS')
plt.show()
```

Result(Captured images)

```
DIS column`s count:  506
DIS column`s mean:  3.795042687747034
DIS column`s std:  2.1057101266276104
```



Description

Use df[‘DIS’].count() to count number of DIS columns.
Use df[‘DIS’].mean() to calculate mean of DIS columns.
Use df[‘DIS’].std() to calculate standard deviation of DIS columns.
alpha is transparency parameter and bins is number of squares.

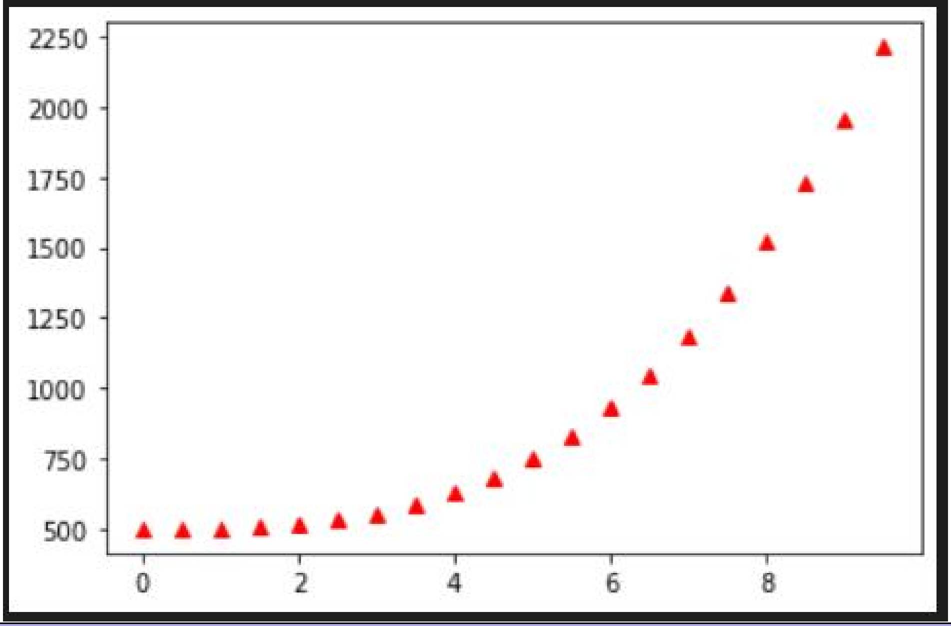
1-4. Matplotlib : Plot $y = 2t^3 + 500$ for t $= [0, 0.5, 1.0, 1.5, \ldots , 8.5, 9.0, 9.5]$ with red triangles.

Code

```python
import numpy as np
import matplotlib.pyplot as plt

t = np.arange(0,9.6,0.5)
y = 2*t**3+500
plt.plot(t,y,'r^')
```

Result(Captured images)

```
[<matplotlib.lines.Line2D at 0x158f2bb75e0>]
```



Description

Use np.arange(start, stop, difference) to set t.
Use plt.plot(x axis, y axis, color and shape)
r^ is red Triangle.

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

Your Answer
Supervised Learning: Using labeled data, Learn a model to predict label (X -> Y). Learning a function that maps an input to an output(label) based on example input-output pairs (training data). Supervised Learning has largely classification and regression. Unsupervised Learning: Using unlabeled data, Learn hidden structure (X1<- X->X2) Learning hidden structures from unlabeled data. Clustering is one of the representative ways of unsupervised learning. Reinforcement Learning: Using reward from actions, Learn action policy (action->reward). Learning how to take actions in an environment in order to maximize the cumulative reward.

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

Your Answer
As learning repeats, the accuracy of learning increases. Ideal learning is that the accuracy increases as data continues to come in and the learning repeats. However, in this process, the phenomenon of overfitting the learning model to the given data, predicting different results even if even a little different data comes in, resulting in lower accuracy is called Overfitting. There is two solutions about overfitting. First solution is reducing feature's number. By reducing the number of features, overfitting will be reduced. However, there is definitely a disadvantage that even features that are important for learning can be discarded in this process. Second solution is using Regularization. Regularization is the process of adding information in order to prevent overfitting.

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
In both gradient descent (GD) and stochastic gradient descent (SGD), you update a set of parameters in an iterative manner to minimize an error function. While in GD, you have to run through ALL the samples in your training set to do a single update for a parameter in a particular iteration, in SGD, on the other hand, you use ONLY ONE or SUBSET of training sample from your training set to do the update for a parameter in a particular iteration. Thus, if the number of training samples are large, in fact very large, then using gradient descent may take too long because in every iteration when you are updating the values of the parameters, you are running through the complete training set. On the other hand, using SGD will be faster because you use only one training sample and it starts improving itself right away from the first sample. SGD often converges much faster compared to GD but the error function is not as well minimized as in the case of GD. Often in most cases, the close approximation that you get in SGD for the parameter values are enough because they reach the optimal values and keep oscillating there.

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)

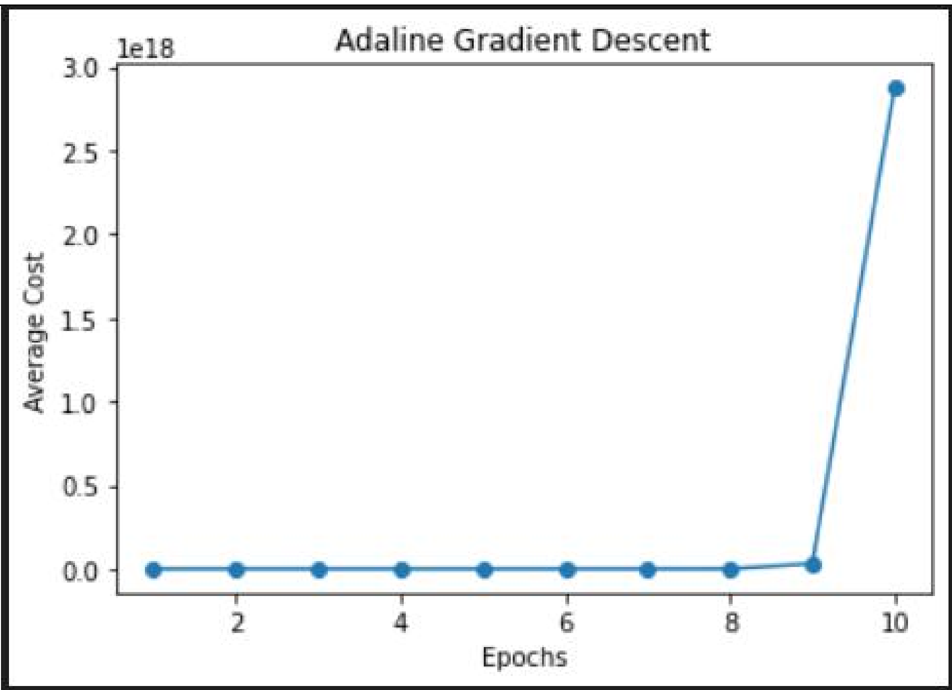
Code

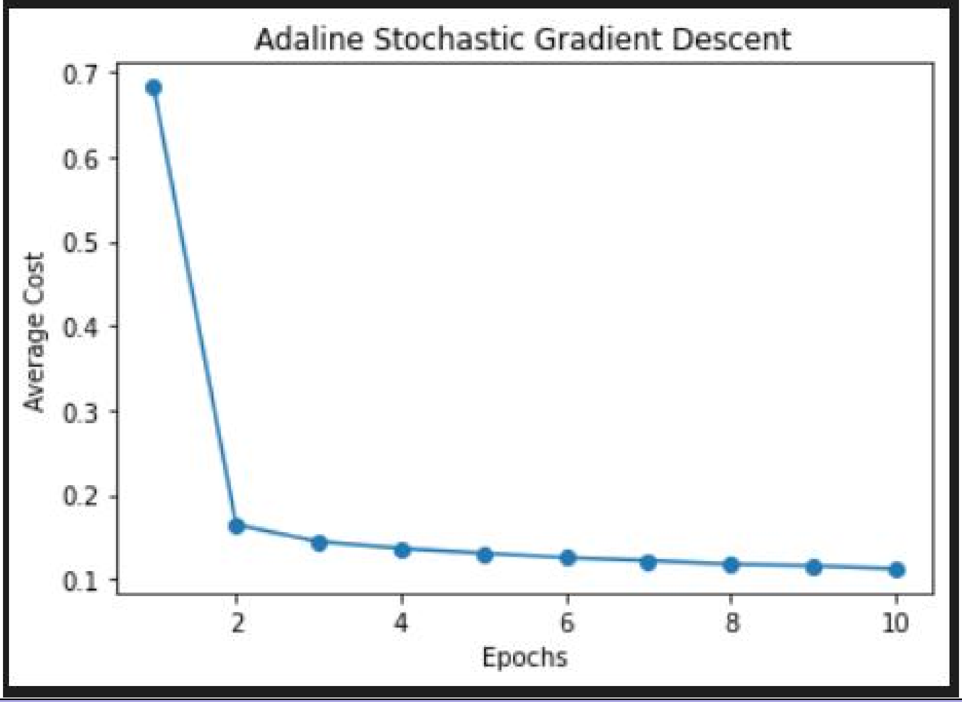
```
####################Adaline Gradient Descent##################################
class AdalineGD(object):
    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        self.eta = eta    # learning rate
        self.n_iter = n_iter  # number of iteration
        self.random_state = random_state

        # weight initiailization
        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 i 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)
            ####################################################

        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)
df = pd.read_csv("./seeds.csv", header = None)
X = df.iloc[0:,[0,1,2,3,4,5,6]].values
y = df.iloc[0:,7].values

X_std = np.copy(X)
X_std[:,0] = (X[:,0]-X[:,0].mean())/X[:,0].std()
X_std[:,1] = (X[:,1]-X[:,1].mean())/X[:,1].std()
X_std[:,2] = (X[:,2]-X[:,2].mean())/X[:,2].std()
X_std[:,3] = (X[:,3]-X[:,3].mean())/X[:,3].std()
X_std[:,4] = (X[:,4]-X[:,4].mean())/X[:,4].std()
X_std[:,5] = (X[:,5]-X[:,5].mean())/X[:,5].std()
X_std[:,6] = (X[:,6]-X[:,6].mean())/X[:,6].std()

ada = AdalineGD(n_iter=10, eta=0.01)
ada.fit(X_std,y)
plt.plot(range(1, len(ada.cost_) + 1), ada.cost_, marker='o')
plt.title('Adaline Gradient Descent')
plt.xlabel('Epochs')
plt.ylabel('Average Cost')
plt.show()
#####################Adaline Gradient Descent###################################
#####################Adaline Stochastic Gradient Descent########################
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 i 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)
```

```
        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)
adas = AdalineSGD(n_iter=10, eta=0.01)
adas.fit(X_std, y)
plt.plot(range(1, len(adas.cost_) + 1), adas.cost_, marker='o')
plt.title('Adaline Stochastic Gradient Descent')
plt.xlabel('Epochs')
plt.ylabel('Average Cost')
plt.show()
#####################Adaline Stochastic Gradient Descent########################
```

Result(Captured images)





Description
Use pd.read_csv() to read seeds.csv file. Use df.iloc[].values to set X and y of seeds.csv file. Standardize features by calculating (X – X`s mean)/X`s standard deviation Set learning rate 0.01 and number of iteration 10.

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