|  |  |
| --- | --- |
| [Machine Learning]  [2021-1] |  |
| Homework 1 |  |
| [Due Date] 2021.04.02  Student ID : 2016112158  Name : KimHeeSu  Professor : Juntae Kim | logo-placeholder |

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

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

|  |
| --- |
| Code |
| 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 |
| Result(Captured images) |
|  |
| Description |
| Circle 클래스의 생성자는 radius를 파라미터로 받고, 그 값으로 self.radius(멤버)를 초기화한다. Circle 클래스의 area메소드는 self.radius로 원의 넓이를 계산하는 메소드이다.  Rectangle 클래스의 생성자는 height(높이)와 (width)를 파라미터로 받고 그 값으로 self.height와 self.width를 초기화한다. area메소드는 self.width와 self.height로 직사각형의 넓이를 계산하는 메소드이다. |

1-2. Numpy : Matrix Dot Product

For , ,

Compute where

Use these functions:

* np.array(), np.arange(),np.dot()
* X.reshape(), X.T

|  |
| --- |
| Code |
| X = np.arange(1,13).reshape((3,4))  w = np.array([0.1,0.2,0.3])  y = X.T.dot(w)  y |
| Result(Captured images) |
|  |
| Description |
| X를 생성할 때 np.arange와 reshape를 사용해서 shape가 (3,4)인 matrix생성  W는 np.array를 이용해 생성한다. X.shape는 (3,4)이고 w.shape는 (3,)이다. shape가 (4,)인 y를 주어진 식대로 계산하려면 matrix dot product를 이용하면 된다. Dot product를 하려면 행과 열을 같게 맞춰줘야 하므로 transpose메소드 T를 사용해 (4,3) x (3,) = (4,)이므로 y.shape와 같다. |

1-3. Pandas : From Boston Housing Price dataset, compute “DIS” column’s count, mean, std. (Week03-Quiz4)

|  |
| --- |
| Code |
| 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()) |
| Result(Captured images) |
|  |
| Description |
| Pandas series는 column내 인스턴스 개수를 계산하는 count, 인스턴스들의 평균을 계산하는 mean, 표준편차를 계산하는 std메소드를 가지고 있다. describe메소드는 count, mean, std뿐만 아니라 min,max, 4분위수들까지 계산해준다 |

1-4. Matplotlib : Plot for with red triangles.

|  |
| --- |
| Code |
| t = np.arange(0, 10, 0.5)  y = 2 \* t\*\*3 + 500  plt.plot(t, y, "r^")  plt.show() |
| Result(Captured images) |
|  |
| Description |
| Np.arange를 이용해서 0~9.5 데이터를 쉽게 생성할 수 있다. 이후, 주어진 식대로 y를 t로 표현한 후, plt.plot의 첫번째 파라미터로 t, 두번째 파라미터로 y를 준 후, 세번째 파라미터로 r^를 주면 알아서 차트가 그려진다. |

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

|  |
| --- |
| Your Answer |
| 지도학습(supervised learning)은 정답이 있는 데이터를 활용해 데이터를 학습시키는 것이다. 입력값(X)가 주어지면 입력값에 대한 Label(Y)를 주어 학습시킨다.  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.  강화학습(Reinforcement learning)은 상과 벌이라는 reward를 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 |

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

|  |
| --- |
| Your Answer |
| 과대적합(overfitting)은 필요이상의 feature를 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 feature를 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 regularization을 가하여 단순화시킨다. |

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 |
| 경사하강법(Gradient Descent, GD)은 배치(단일 반복에서 기울기를 계산하는데 사용하는 샘플의 총 개수)를 전체 데이터 셋(batch\_size==len(train\_set))으로 두고 그래디언트를 계산하여 그래디언트 값에 따라 weight를 업데이트 하는 방법이다. 그에 반해, SGD는 반복당 하나의 샘플(즉, 배치 크기가 1)에 대해서 그래디언트를 계산한 후 weight를 업데이트하는 방법이다. GD는 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. SGD는 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 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)  ####################################################  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 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)  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 i in range(7):  X\_std[:,i] = (X\_std[:,i] - X\_std[:,i].mean()) / X\_std[:,i].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(ada.cost\_) + 1), adaSGD.cost\_, marker='o')  plt.xlabel('Epochs')  plt.ylabel('Average Cost')  plt.show() |
| Result(Captured images) |
|  |
| Description |
|  |

**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