CASE STUDY 2:

REGRESSION – Boston Housing Data

\*\*\*

Nhóm thực hiện: 6

1. Nguyễn Thùy Linh|17520689
2. Nguyễn Hữu Phong|17520015
3. PHÁT BIỂU BÀI TOÁN
4. **Mục đích**

* Xây dựng mô hình dự đoán giá nhà ở Boston dựa trên các phương pháp regression đã được học:

+ Gradient Descent : Batch, Stochastic, Mini-Batch

+ Normal Equations : Basic, Centroid

* Đánh giá các phương pháp đó.

1. **Đầu vào**

Một dataset dự đoán giá nhà gồm 506 tập giá trị quan sát với 14 thuộc tính, trong đó MEDV là thuộc tính class:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Kí hiệu | Miêu tả - EL | Miêu tả - VN |
| 1 | CRIM | per capita crime rate by town | Tỉ lệ tội phạm bình quân đầu người theo thị trấn |
| 2 | ZN | proportion of residential land zoned for lots over 25,000 sq.ft | Tỉ lệ đất ở chia thành các lô trên mỗi 25,000 ft vuông |
| 3 | INDUS | proportion of non-retail business acres per town | Tỉ lệ mẫu đất kinh doanh không bán lẻ trên mỗi thị trấn |
| 4 | CHAS | Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) | = 1 nếu giáp với sông Charles  = 0 ngược lại |
| 5 | NOX | nitric oxides concentration (parts per 10 million) | Nồng độ oxit nitric(1/10 triệu) |
| 6 | RM | average number of rooms per dwelling | Số phòng trung bình trên mỗi căn hộ |
| 7 | AGE | proportion of owner-occupied units built prior to 1940 | Tỉ lệ nhà-ở-của-chủ- sở-hữu-không-cho-thuê được xây trước 1940 |
| 8 | DIS | weighted distances to five Boston employment centres | Khoảng cách tới 5 trung tâm làm việc ở Boston |
| 9 | RAD | index of accessibility to radial highways | Chỉ số tiếp cận đến các đường cao tốc xuyên tâm |
| 10 | TAX | full-value property-tax rate per $10,000 | Tỉ lệ thuế tài sản full-value trên $10,000 |
| 11 | PTRATIO | pupil-teacher ratio by town | Tỉ lệ học sinh-giáo viên mỗi trị trấn |
| 12 | B | 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town | = 1000(Bk – 0.63)2 với Bk là tỉ lệ người da đen mỗi trị trấn |
| 13 | LSTAT | % lower status of the population | % dân số thuộc tầng lớp hạ lưu |
| 14 | MEDV | Median value of owner-occupied homes in $1000's | Giá trung bình các ngôi nhà sở-hữu-không-cho-thuê trong tầm $1000 |

1. **Đầu ra**

* Mô hình dự đoán giá nhà ở Boston.

1. **Cách đánh giá mô hình**

* Sử dụng MSE - Mean Square Error (Sai số trung bình)

1. XÂY DỰNG MÔ HÌNH TỪ TẬP HUẤN LUYỆN SỬ DỤNG THUẬT TOÁN BGD – Batch Gradient Descent
2. **Trình bày thuật toán dưới dạng mã giả**

def BGD(dataset, w\_init, gamma, epsilon):  
 w = w\_init  
 for i in range(len(dataset))://số lần lặp = chiều dài mẫu

E = MSE (w)   
 G = Gradient(w)  
 w\_new = w – gamma\*G //cập nhật w  
 if distance(w,w\_new) < epsilon:  
 return w\_new  
 w = w\_new  
 return w

1. **Minh họa thuật toán trên Boston Housing Data**

Minh họa 2 lần lặp với 5 mẫu dữ liệu đầu tiên:

class

1. 0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
2. 0.02731 0.00 7.070 0 0.4690 6.4210 78.90 4.9671 2 242.0 17.80 396.90 9.14 21.60
3. 0.02729 0.00 7.070 0 0.4690 7.1850 61.10 4.9671 2 242.0 17.80 392.83 4.03 34.70
4. 0.03237 0.00 2.180 0 0.4580 6.9980 45.80 6.0622 3 222.0 18.70 394.63 2.94 33.40
5. 0.06905 0.00 2.180 0 0.4580 7.1470 54.20 6.0622 3 222.0 18.70 396.90 5.33 36.20

* Lặp lần 1:
* Khởi tạo w = [w0 ,w1 ,..., w13 ] = [0]
* f(w)i = w0 + w1. x1 + ... + w13 . x13 = 0
* Tính MSE = 934.53
* Tính gradient [dw0 ,dw1 ,..., dw13 ] và

Cập nhật w = [w0 ,w1 ,..., w13 ]:

|  |  |
| --- | --- |
| **Gradient** | **w** |
| -59.96 | 0.5996 |
| -2.1077228 | 0.021077228 |
| -172.8 | 1.728 |
| -242.0836 | 2.420836 |
| 0 | 0 |
| -28.4774 | 0.284774 |
| -415.30708 | 4.1530708 |
| -3552.388 | 35.52388 |
| -319.89474 | 3.1989474 |
| -138.16 | 1.3816 |
| -14471.92 | 144.7192 |
| -1068.344 | 10.68344 |
| -16544.24376 | 165.4424376 |
| -299.1708 | 2.991708 |

* Lặp lần 2:
* MSE = 6791437856

|  |  |
| --- | --- |
| **Gradient** | **w** |
| 154573.5554 | -1545.135954 |
| 5703.805572 | -57.01697849 |
| 331511.4519 | -3313.386519 |
| 615469.1714 | -6152.270878 |
| 0 | 0 |
| 72889.16671 | -728.6068931 |
| 1078592.066 | -10781.76759 |
| 8912334.871 | -89087.82483 |
| 838900.459 | -8385.805642 |
| 370422.9578 | -3702.847978 |
| 36807471.75 | -367929.9983 |
| 2777089.863 | -27760.21519 |
| 47767683.26 | -477511.3902 |
| 728305.9578 | -7280.06787 |

1. THỰC NGHIỆM ĐÁNH GIÁ
2. **Cách thức thực nghiệm, đánh giá:**

* **Cách chia Data set thành Train set và Test set**
* *Thủ công:*
* Chia tập dữ liệu thu thập được thành 2 phần theo tỉ lệ 9:1

+ Training set: 455 mẫu

+ Test set: 51 mẫu

* Cách chia: Duyệt từ đầu đến cuối tập data, lấy Test set với bước nhảy là 10
* *Thư viện:*
* Chia tập dữ liệu thu thập được thành 2 phần theo tỉ lệ 9:1 sử dụng hàm train\_test\_split() của thư viện sklearn:

+ Training set: 455 mẫu

+ Test set: 51 mẫu

* **Cách đánh giá:**
* Dùng công thức MSE – Mean Square Error:

MSE =

1. **Kết quả đánh giá:**

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **MSE** | **Những điểm hạn chế** |
| **BGD** | 72.29908924208212 | Hạn chế đối với cơ sở dữ liệu có vô cùng nhiều điểm, việc tính toán lại đạo hàm với tất cả các điểm này sau mỗi vòng lặp trở nên cồng kềnh và không hiệu quả. |
| **Normal Equations**  **(basic)** | 20.544274659320653 | Chỉ sử dụng được khi (XT.X) khả nghịch. |

1. LẬP TRÌNH CÀI ĐẶT
2. **BGD – Batch Gradient Descent**

* **Các tool, thư viện đã sử dụng:**
* Không có.
* **Cách cài đặt tool và chạy chương trình:**
* Đầu tiên ta đọc file và cho dataset vào mảng data.
* Ta tiếp tục phân chia mảng data thành mảng train và test theo tỉ lệ 9:1
* Lấy learning rate = epsilon = 0.000001 và khởi tạo w = {0}
* Xây dựng các hàm tính toán ***f(w,row)***, ***MSE(set, w)***, ***Gradient(w)*** và điều kiện dừng ***isConstant(w, w\_new).***
* Cuối cùng ta định nghĩa hàm ***BGD()*** để tìm ra bộ trọng số w và tính MSE khi áp dụng bộ trọng số đó lên test set.
* **Source code:**
* def readFile(path):  
   file = open(path,'r')  
   data = [line.split() for line in file]  
   for row in data:  
   for i in range(len(row)):  
   row[i] = float(row[i])  
   return data  
    
  data = readFile('Dataset')  
    
  gamma = 0.000001  
  epsilon = 0.000001  
    
  def split(data):  
   train = []  
   test = []  
   for i in range(0,len(data),10):  
   test.append(data[i])  
   for j in range(i+1,i+10):  
   if j < len(data):  
   train.append(data[j])  
   return train,test  
    
  train , test = split(data)  
    
  def f(w,row):  
   fx = w[0]  
   for i in range(len(row)-1):  
   fx += w[i+1]\*row[i]  
   return fx  
    
  def MSE(set,w):  
   E = 0  
   for row in set:  
   E += (row[len(row)-1]-f(w,row))\*\*2  
   return E/len(set)  
    
  def Gradient(w):  
   G = [0]\*14  
   for i in range(len(G)):  
   if i == 0:  
   for row in train:  
   G[i] += row[len(row)-1]-f(w,row)  
   else:  
   for row in train:  
   G[i] += row[i-1]\*(row[len(row)-1]-f(w,row))  
   for i in range(len(G)):  
   G[i] \*= (-2/len(train))  
   return G  
    
  def isConstant(w, w\_new):  
   for i in range(len(w)):  
   if abs(w\_new[i] - w[i]) > epsilon:  
   return False  
   return True  
    
  def BGD():  
   w = [0]\*14  
   E = 0  
   for i in range(len(train)):  
   E = MSE(train,w)  
   G = Gradient(w)  
   w\_new = []  
   for j in range(len(w)):  
   w\_new.append(w[j] - gamma\*G[j])  
   if isConstant(w,w\_new) == True:  
   return w\_new  
   w = w\_new  
   return w,E  
    
  w,E = BGD()  
  print('W = ',w)  
  print('MSE (train) = ',E)  
  print('MSE (test) = ',MSE(test,w))
* **Kết quả trả về của hệ thống:**
* W = [0.0007200333955247214, -0.007443776317927609, 0.0459795202804453, -0.0062041219296919995, 0.0003341957115597785, 0.00022705958778893568, 0.008114912733399645, 0.004766003920949037, 0.0034981319582702056, -0.0019464886402587069, 5.8172853570684665e-05, 0.006179639641520317, 0.05961974563241947, -0.0200741080260589]  
  MSE (train) = 76.3787185755272  
  MSE (test) = 72.29908924208212

1. **Normal Equations – basic**

* **Các tool, thư viện đã sử dụng:**
* numpy
* sklearn
* **Cách cài đặt tool và chạy chương trình:**
* Đầu tiên ta đọc file và cho dataset vào mảng data.
* Chuẩn hóa data – thêm data [i][0] = 1
* Tách data thành X - datapoints và y – target.
* Sử dụng hàm ***train\_test\_split*** của ***sklearn.model\_selection*** để chia data thành train và test set
* Xây dựng các hàm ***NE\_basic()*** để tìm bộ trọng số w, ***MSE(n, y, X)*** để tính MSE cho train và test khi áp dụng w .
* **Source code:**

import numpy as np  
from sklearn.model\_selection import train\_test\_split  
  
def readFile(path):  
 file = open(path,'r')  
 data = [line.split() for line in file]  
 for row in data:  
 for i in range(len(row)):  
 row[i] = float(row[i])  
 return data  
data = readFile('Dataset')  
  
def normalize():  
 for row in data:  
 row.insert(0,1)  
normalize()  
  
def split():  
 X = []  
 y = []  
 for row in data:  
 temp = []  
 for i in row:  
 if i is row[14]:  
 y.append(i)  
 else:  
 temp.append(i)  
 X.append(temp)  
 return X,y  
X,y = split()  
  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.1,random\_state=1)  
X\_train,X\_test,y\_train,y\_test = np.array(X\_train),np.array(X\_test),np.array(y\_train),np.array(y\_test)  
  
def NE\_basic():  
 A = (X\_train.T).dot(X\_train)  
 A\_inverted = np.linalg.inv(A)  
 b = (X\_train.T).dot(y\_train)  
 return A\_inverted.dot(b)  
  
w = NE\_basic()  
  
def MSE(n,y, X):  
 return (np.linalg.norm(y - X.dot(w),2))\*\*2/n  
  
E\_train = MSE(X\_train.shape[0],y\_train,X\_train)  
E\_test = MSE(X\_test.shape[0],y\_test,X\_test)  
  
print('W = ',w)  
print('MSE (train) = ',E\_train)  
print('MSE (test) = ',E\_test)

* **Kết quả trả về của hệ thống:**

W = [ 3.89353296e+01 -1.16259926e-01 5.56061815e-02 2.42066570e-03  
 2.58498034e+00 -1.91474547e+01 3.54362973e+00 -5.86296897e-04  
 -1.59321860e+00 3.16227116e-01 -1.20965602e-02 -9.20798781e-01  
 8.75217941e-03 -5.18191990e-01]  
MSE (train) = 22.127472016066708  
MSE (test) = 20.544274659320653