به نام خدا

گزارش کار تمرین سری دوم درس یادگیری ماشین

پیاده سازی رگرسیون لاجیستیک

سینا دالوند

40011415053

ابتدا توسط کتابخانه پانداس اطلاعات فایل اکسل را خوانده و در متغییر df واریز میکنیم.

```
import pandas as pd
import random

df = pd.read_excel('dataset.xls', 'Data').to_numpy()
```

حال لازم است کاملا به صورت رندوم نسبت ۷۰ به ۳۰ داده های خوانده شده را جدا سازی کنیم که ۷۰ درصد آن در مجموعه آموزشی و ۳۰ درصد آن در مجموعه تست قرار میگیرد ، برای انجام این عملیات ابتدا آرایه ای از Trueو False ها به اندازه کل دیتاست و به نسبت خواسته شده ایجاد میکنم و آن را کاملا به طور رندوم به هم میریزم سپس با توجه به داده های تولید شده در صورتی که آیتم متناظر با آیتم دیتا ست مقدار Trueداشت آن را در دسته آموزشی و غیر این صورت در دسته تست قرار میدهیم.

```
Tuning data (70% Training, 30% Testing)

threshold = int(len(df) * 0.7)
randomArray = [True if i < threshold else False for i in range(len(df))]
random.shuffle(randomArray)
randomArray = list(zip(randomArray, df.tolist()))
test_set = np.array(list(map(lambda y: y[1], filter(lambda x: x[0] == False, randomArray))))
train_set = np.array(list(map(lambda y: y[1], filter(lambda x: x[0] == True, randomArray))))
print(f"train_set size: {len(train_set)}, test_set size: {len(test_set)}")
```

اگر به دیتا ست توجه کنیم متوجه میشویم که مقادیر فیچر ها دارای رنج متفاوت میباشد که این امر در هنگام محاسبه رگرسیون باعث ایجاد مشکل میشود و لازم است توزیع نرمال بر روی داده ها در نظر گرفته شود به همین دلیل با نرمالایز کردن فیچر ها آنها را در رنج عددی ۰ تا ۱ قرار میدهیم .

```
now should do scale on data

def normalize(data):
    data = data.astype(float)
    return (data - np.mean(data,axis=0)) / np.std(data, axis=0)

X_train = normalize(X_train)
X_test = normalize(X_test)
X_train = np.column_stack((np.ones((len(X_train), 1)), X_train))
X_test = np.column_stack((np.ones((len(X_test), 1)), X_test))
```

حال زمان محاسبه مینیم مقادیر برای تتا است ، بدین منظور از گرادیان کاهشی استفاده میکنیم که متد آن به شکل زیر پیاده سازی شده است :

```
1  def gradientDescent(X, y, alpha, itr):
2      n = X.shape[1]
3      classes = list(set(y))
4      thetas = []
5      for j in range(len(classes)):
6          theta = np.zeros((n, 1))
7          innerY = np.asarray(list(map(lambda x: [1] if (x[0] == classes[j]) else [0], y)))
8          m = len(innerY)
9          for i in range(itr):
10             theta = theta - (alpha * (1 / m * (np.dot(X.T, (sigmoid(np.dot(X, theta)) - innerY)))))
11         thetas.append(theta.flatten())
12     return thetas, classes
13
```

شیوه کارکرد گرادیات کاهشی لاجستیک همانند گرادیان کاهشی خطی است اما در این بین تفاوت هایی وجود دارد که به توضیح آنها میپردازیم :

- تابع h(x) متفاوت است.
این تابع در الگوریتم لاجستیک با استفاده از تابع سیگموید تعریف میشود که میتوان شیوه پیاده سازی آن را در خط ۱۰ مشاهده کرد.

- به دلیل multinominal بودن مسئله بر خلاف حالت باینری ما بیش از ۲ حالت را داریم که بدین منظور لازم است الگوریم برای هر کدام از کلاس ها یکبار به طور مجزا اجرا شود که حلقه خط ۵ در تصویر بالا بدین منظور میباشد و سپس با استفاده از این حلقه در خط ۷ هربار یک کلاس دریافت شده و سایر کلاس ها را به دید یکسان نگاه میکنید و به نوعی به صورت باینری به حل مسئله و محاسبه مقادیر تتاها برای هر کلاس میپردازد.

خروجی متد بالا یک آرایه چند بعد از مقادیر تتا ها برای هر حالت است و همچنین لیبل متنی کلاس ها به علت تبدیل آنها به صورت دسیمال جهت شناسایی کلاس ها در آینده همراه آن ارسال میشود.

حال میتوان گفت مقادیر مدل ما آماده و نیازمند تست جهت بررسی دقت مدت می باشد.

بررسی دقت مدل تولید شده :

```
Evaluate the model

1  def predict(X, theta):
2      p = sigmoid(X @ np.asarray(theta).T)
3      p = np.asarray(list(map(lambda x: np.argmax(x), p)))
4      return p
5
6  train_set_value = list(map(lambda x: classes[x], predict(X_train, thetas)))
7  train_set_percent = sum(train_set_value == Y_train) / len(Y_train)
8  print(f"Accuracy for Train Set: {train_set_percent}")
9
10 test_set_value = list(map(lambda x: classes[x], predict(X_test, thetas)))
11 test_set_percent = sum(test_set_value == Y_test) / len(Y_test)
12 print(f"Accuracy for Test Set: {test_set_percent}")

  Accuracy for Train Set: 1.0
  Accuracy for Test Set: 0.9814814814814815
```

حال برای بررسی صحت کارکرد مدل ، نیاز است که با اعمال مقادیر تتا در h(x) و وارد کردن فیچر ها مجموعه تست در آن ، بررسی کنیم خروجی حاصل چه مقدار با Y های مجموعه تست تشابه دارد.

بدین منظور تابعی تحت عنوان predict تعریف کردیم که در پارامتر اول داده هایی که قرار است بر روی آنها پیشبینی انجام شود و در پارامتر دوم تتا های حاصل از مرحله قبل را بدان پاس میدهیم.

در خط ۶ تا ۸ ابتدا بررسی بر روی داده های ترین که خود مدل به واسطه آن آموزش دیده را میدهیم ومیبینم که خروجی مقدار ۱ را میدهید و بدین معناست که مدل دقت ۱۰۰ درصدی را بر روی مجموعه آموزشی دارد ، حال برای به چالش کشیدن داده های دیده نشده را به مدل میدهیم که شامل خطوط ۱۰ تا ۱۲ میباشد که خروجی نیز دارای دقت ۹۸ درصدی است .

پس مدل ما به خوبی آموزش دیده و دقت پیشبنی آن ۹۸ درصد درست است.