گزارش تمرین پرسپتورون

یاسمین رحیمی ۸۱۰۸۹۶۰۲۱

تعریف سوال

یک سری کاراکتر داریم (A,B,C,D,E,J,K) و یک سری داده برای یادگیری(از هر کاراکتر ۳ داده ی مجزا داریم که هر کدام به صورت یک ماتریس ۹*۷ نشان داده شده) . حال می‌خواهیم به وسیله‌ی الگوریتم پرسپتورون وزن شبکه‌های عصبی را با استفاده از این داده‌ها تغیر بدهیم تا به مقداری برسیم که شبکه بتواند با تقریب خوبی داده‌های یادگیری و داده‌های تست را درست تشخیص دهد.

الگوریتم پرسپتورون

در الگوریتم پرسپتورون ما به صورت زیر عمل می کنیم:

مرحله ی اول:

به وزن‌ها و بایاس و نرخ یادگیری مقدار می دهیم. برای سادگی می‌توانیم نرخ یادگیری را ۱ و باقی را ۰ در نظر بگیریم.

مرحله ی دوم:

به ازای ورودی هر کاراکتر اکتیویشن فانکشن به این صورت عمل می‌کند:

$$y\,in = b + \sum_i x_i w_i;$$

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \le y\_in \le \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

مرحله ی سوم:

وزن ها به صورت زیر آپدیت می‌شوند:

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i,$$

$$b(\text{new}) = b(\text{old}) + \alpha t.$$

هر زمانی که وزن‌های قدیمی بعد از یک اپیک تغیری نکند و ثابت بماند مقداردهی وزن‌ها به اتمام رسیده و برنامه مرحله‌ی یادگیری را به پایان می‌رساند.

پیاده‌سازی

من کد این الگوریتم را در زبان پایتون نوشتم و خلاصه ای از تابع عملکرد آن را در زیر توضیح می‌دهم:

```
def Readfile():
    train_data = []
    with open("training-data.txt") as file:
        cnt,ind = 0,0
        train = num.zeros(64, dtype=int)
        for line in file:
            if cnt == 0:
                train[0] = 1
                ind+=1
                cnt+=1
                for element in line:
                    if element=='#':
                        train[ind] = 1
                        ind+=1
                    elif element=='*':
                        train[ind] = -1
                        ind+=1

            elif cnt==9:
                cnt,ind = 0,0
                train_data.append(train)
                train = num.zeros(64,dtype=int)
            else:
                for element in line:
                    if element=='#':
                        train[ind] = 1
                        ind+=1
                    elif element=='*':
                        train[ind] = -1
                        ind+=1
                cnt+=1
        return train_data
```

تابع **Readfile()**

در این تابع من ۲۱ ماتریس کاراکترها (از هر کاراکتر ۳ نمونه وجود دارد) را در یک فایل تکست می‌گیرم و مقادیر هر ۹ خط فایل را (یعنی هر کاراکتر را) در یک آرایه train ذخیره می‌کنم. این آرایه از پکیج numpy تعریف شده که استفاده از اعضای این پکیج برای انجام کارهای محاسباتی توصیه میشود. تمام این آرایه ها به لیست train_data افزوده می‌شوند.

```
def Actfunc (y_in):

    if y_in > 0:
        y_out = 1
    else:
        y_out = -1
    return y_out
```

تابع **Actfunc**

این تابع کار اکتیویشن فانکشن را انجام می‌دهد. در این برنامه من تتا را برابر صفر قرار دادم.

تابع **Learn()**

این تابع لیست داده های یادگیری را به عنوان ورودی می‌گیرد و در یک حلقه به تعداد کاراکترها هر بار یک آرایه‌ی ۶۴ تایی وزن شبکه تولید می‌شود و سپس وزن مرتبط با هر پرسپترون در آرایه‌ی مربوط به کاراکتر آن ضرب داخلی می‌شود و تابع Actfunc(Activation function) فراخوانی می‌شود. مقداری که تابع برگرداند اگر با تارگت ما همخوانی نداشت وزن متناظر آپدیت می‌شود تا زمانی که وزن‌ها به گونه‌ای شود که کاراکتر مورد نظر برای پرسپترون متناظرش خروجی ۱ بدهد و برای باقی پرسپترون‌ها خروجی 1- بدهد. در این زمان وزن‌ها دیگر تغییری نمی‌کنند و تعداد دفعاتی که وزن‌ها تغیر کرده تا به ثبات برسد در پارامتر اپیک ذخیره می‌شود.

```
def Learn(train_data):
    epoch = 0
    weights = []
    while True:
        for j in range(0,7):
            weight = num.random.rand( 64,)
            backstep = weight.copy()
            for i in range (0, len(train_data)):
                    s = train_data[i]
                    y_in = weight.dot(s)
                    Actfunc(y_in)
                    if i%7 == j:
                        target = 1
                    else :
                        traget = -1
                    if y_in != target :
                        weight += target * train_data[i]
            epoch += 1
            print(weight)
            if num.array_equal(backstep, weight):
                break
            weights.append(weight)
        break
    print (weights)
    return weights
```

test()

تابع تست به این صورت عمل می‌کند که یک کاراکتر را به عنوان ورودی می‌گیریم و ماتریس وزن آن را بسته به این که ورودی چه حرفی باشد روی آن اجرا می‌کنیم. در صورتی که اکتیویشن فانکشن آن برابر ۱ شود برنامه اعلام می‌کند که به درستی تشخیص داده شده ولی اگر با ماتریس وزن حداقل یک کاراکتر دیگر هم خروجی ۱ بدهد اعلام می‌کنیم که الگوریتم شکست خورده.

```
def test (weights):
    character = input("Please enter character: ")
    if character == "A":
        targ = 0
    elif character == "B":
        targ = 1
    elif character == "C":
        targ = 2
    elif character == "D":
        targ = 3
    elif character == "E":
        targ = 4
    elif character == "J":
        targ = 5
    elif character == "K":
        targ = 6
    with open("test-data.txt") as file:
        test = num.zeros(64, dtype=int)
        ind = 1
        test[0]== 1
        for line in file:
            for element in line:
                if element=='#':
                    test[ind] = 1
                    ind+=1
                elif element=='*':
                    test[ind] = -1
                    ind+=1
        testweight = num.zeros((64,1), dtype=int)
        for j in range (0, 7):
            testweight = weights[j]
            y_in = testweight.dot(test)
            if Actfunc (y_in) == 1 and j == targ:
                print("Perceptron recognized character")
            elif Actfunc (y_in) == 1 and j != targ:
                print (j)
                print("Perceptron failed")
            elif Actfunc (y_in) == -1 and j == targ:
                print("Silly perceptron")
            #print (weight)
```

اجرا:

برنامه را اجرا می کنیم و وزن های تولید شده را پرینت می کنیم که به شرح زیر است:
این آرایه ی وزن کاراکتر اول یعنی A است :

```
[array([[-0.53854187, -2.00237055, -0.4531011 ,  1.57440331,  1.18113521,
        -0.96430305, -2.41497613,  1.28571804, -0.69595687, -0.30306653,
         1.72244994,  3.97130072, -0.21905924,  1.21700698, -0.25772754,
        -0.08656847, -0.9525952 ,  1.71168579,  1.35278152,  1.06978057,
         1.93305216, -0.84076019, -0.90737084, -0.99713458,  1.79111691,
         1.9637444 ,  3.92300614,  1.20258023, -0.87927284, -0.61127927,
        -2.72668359,  1.97781361, -0.66733887,  1.21245031, -0.51161704,
         1.04971442, -0.03157202,  1.58117311,  1.07735604,  3.65806517,
         3.22030032,  3.1892683 , -0.64979642, -0.21263537,  1.23870189,
         1.16702446, -0.59496603,  1.0190751 ,  3.13227319, -0.23685312,
        -0.52528825,  1.00764447,  1.14783639,  1.73608495, -0.59297633,
         3.92105903, -0.78569002, -0.3200797 ,  1.91998205,  1.9868401 ,
        -0.80763693,  1.94797861, -0.83570617,  3.33032807]]), array([[-0.604092
```

و حالا برنامه تابع تست را روی فایل تست اجرا می کند که در بردارنده ی یک A نویزی است و خروجی نشان می دهد برنامه ی ما تشخیص داده که ورودی همان است ولی به عنوان E هم کاراکتر ما را می پذیرد

```
Perceptron recognized character
4
Perceptron failed
```