به نام خدا

پروژه سوم درس هوش محاسباتی

یادگیری روابط دروازه‌های منطقی XOR، AND و OR

نام دانشجو:

امیرحسین شریفی‌صدر ۹۷۳۳۰۴۴

استاد: دکتر سیدصالحی

تدریسیار: خانم خلیلی

پاییز ۱۴۰۰

مقدمه:

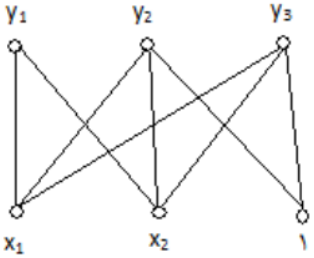
در این تمرین هدف یادگیری دروزاه‌های منطقی AND، OR و XOR است با استفاده از شبکه مصنوعی پرسپترون و الگوریتم یادگیری LMS است.

۲ ورودی برای هر شبکه داریم و سه خروجی نیز که همان گیت‌های ما هستند مدنظر است.

۱ ورودی بایاس نیز وجود دارد که عملا با آن کاری نخواهیم داشت.

روند کلی به شکل زیر است:

یک شبکه عصبی یک لایه وجود دارد که با دو ورودی به ما سه خروجی خواهد داد و سپس خروجی توسط تابع فعالسازی به خروجی مدنظر ما که باینری است تبدیل میشود، دراینجا تابع فعالسازی ما تابع پله است:

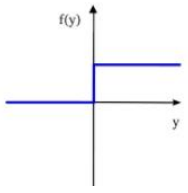


$$f(y)=\begin{cases}1 & y\geq 0\\ 0 & y<0\end{cases}$$

و به شکل ماتریسی و ریاضی نیز الگوریتم به شیوه زیر است:

$$\begin{bmatrix}0&0&1\\0&1&1\\1&0&1\\1&1&1\end{bmatrix}\times\begin{bmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\\\theta_1&\theta_2&\theta_3\end{bmatrix}=\begin{bmatrix}\theta_1&\theta_2&\theta_3\\w_{21}+\theta_1&w_{22}+\theta_2&w_{23}+\theta_3\\w_{11}+\theta_1&w_{12}+\theta_2&w_{13}+\theta_3\\w_{11}+w_{21}+\theta_1&w_{12}+w_{22}+\theta_2&w_{13}+w_{23}+\theta_3\end{bmatrix}$$

$$f(y)=\begin{cases}1 & y\geq 0\\ 0 & y<0\end{cases}$$



در ادامه گزارش بخش به بخش مراحل همراه با کد مربوط توضیح داده خواهد شد.

کتاب‌خانه‌های استفاده شده:

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

بخش اول:

نحوه ساخت و نمایش تابع ورودی:

```
x = np.array([[0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]])
print(x)
```

```
[[0 0 1]
 [0 1 1]
 [1 0 1]
 [1 1 1]]
```

نحوه ساخت مارتیس وزن‌ها:

```
n = 0.001
w = n* np.random.random(size=(3, 3))
print(w)
```

```
[[0.00032906 0.00021539 0.00048664]
 [0.00033366 0.0008073  0.00023096]
 [0.00074321 0.00082872 0.00053481]]
```

بخش دوم:

در این بخش مانند پروژه قبل با استفاده از الگوریتم LMS خروجی را تولید میکنیم. این الگوریتم مربعات خطای بین خروجی به دست آمده و خروجی مطلوب را محسابه میکند و با کاهش آن ما را به خروجی مطلوب نزدیک میکند.

به این شکل که در هر مرحله ماتریس ورودی ما در ماتریس وزن ما ضرب میشود و خروجی را به ما میدهد. البته در این مرحله خروجی ما حالت باینری ندارد و عدد حقیقی است که آنرا با استفاده از تابع فعالسازی که در اینجا تابع پله است تبدیل به حالت باینری میکینم. بدین شکل که هر جا خروجی کمتر از صفر بود به صفر تبدیل شود و هرجا خروی بزگتر و یا مساوی صفر بود به ۱ تبدیل شود.

حال در مرحله بروزرسانی وزن‌ها با استفاده از الگوریتم LMS فرمول زیر را اجرا میکنیم:

$$\mathrm{w}(n+1) = w(n) + \eta x^{T}(n)\mathrm{e}(\mathrm{n})$$

که n در آن ضریب یادگیری است که مقدارش را همان ۰.۰۰۱ قرار دادیم. کوچکتر بودنش سبب کاهش سرعت همگرایی به خروجی مطلوب و در عوض افزایش صحت یادگیری میشود.

این مقدار گذاشته شده برای n مقدار مناسبی پیشبینی میشود.

تعریف مارتیس خروجی و ماتریسی که خطای بین خروجی مطلوب و خروجی به دست آمده را نشان میدهد و همچنین ماتریس خطای شبکه در هر مرحله از تکرار برای هر گیت منطقی:

۴۰ بار تکرار در این برنامه انجام شده است.

```
y = np.zeros((4,3))
e = np.zeros((4,3))
e_mean = np.zeros((40,3))
```

تعریف خروجی مطلوب و مدنظر شبکه :

```
d = np.array([[0,0,0],[0,1,1],[0,1,1],[1,1,0]])
```

نحوه محاسبه خروجی نورون‌ها:

```
for i in range(0,40):
    y = np.dot(x,w)
    for j in range(0,4):
        for k in range(0,3):
            if y[j,k]>= 0 :
                y[j,k] = 1
            if y[j,k]<0:
                y[j,k] = 0
    x1 = np.transpose(x)
    e = d - y
    w = w + n*np.dot(x1,e)
```

به تفکیک در خط اول ۴۰ بار تکرار برای یادگیری شبکه در نظر گرفته شده است یعنی ۴۰ بار تمامی کدهای پایین تکرار میشوند.

در خط دوم خروجی با استفاده از ضرب ماتریس ورودی در وزن محسابه شده است.

در خط ۳ تا ۸ ماتریس خروجی ما با استفاده از تابع پله به حالت باینری که مدنظر ما است در آمده.

و در ۳ خط آخر نیز عملیات برزورسانی وزن انجام شده است. بدین شکل که خطای بین خروجی مطلوب و خروجی به دست آمده محسابه شده و ماتریس ترانهاده ورودی‌ها نیز محاسبه میشود و در فرمول بروزرسانی وزن قرار داده میشوند.

بخش سوم:

تشکیل ماتریس و درنهایت منحنی خطای شبکه در هر تکراربرای هر سه خروجی OR، AND و XOR:

```
# calculating error for three gates: or, and, xor
e_mean[i, 0] = abs(np.mean(e[0:4 , 0]))
e_mean[i, 1] = abs(np.mean(e[0:4 , 1]))
e_mean[i, 2] = abs(np.mean(e[0:4 , 2]))
```

مشاهده میشود که میانگین و قدر مطلق خطا برای هر دروازه در هر مرحله محاسبه میشود.

تعداد تکرار مراحل ۴۰ بار است پس ما ۴۰ ردیف و ۳ ستون خواهیم داشت.

بخش چهارم:

مشاهده خروجی شبکه و خطای شبکه در هر تکرار:

```
# y after learning in neural network
print(y)
# printing error for OR,AND, XOR gates
print(e_mean)
```

```
[[0. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 1. 0.]]
```

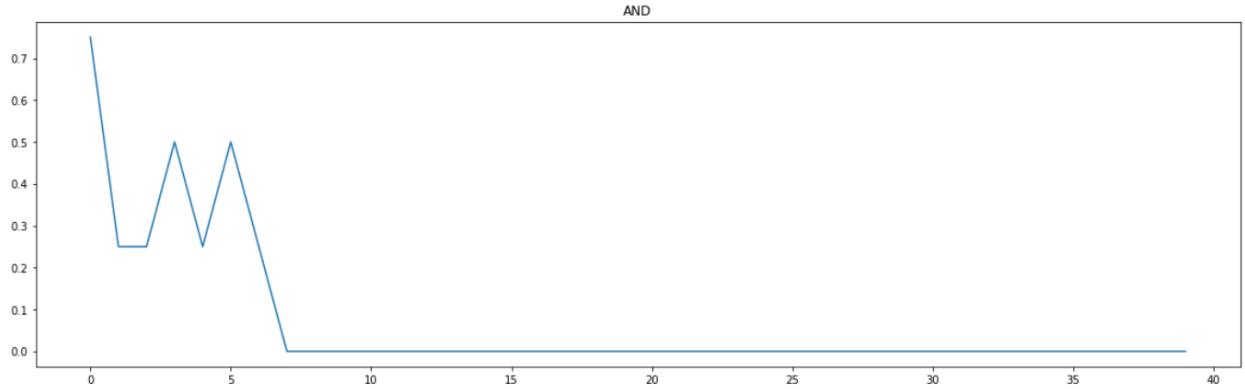
خروجی شبکه:

```
 [[0.75 0.25 0.5 ]
  [0.25 0.   0.5 ]
  [0.25 0.   0.5 ]
  [0.5  0.   0.5 ]
  [0.25 0.   0.5 ]
  [0.5  0.   0.5 ]
  [0.25 0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]
  [0.   0.   0.5 ]]
```

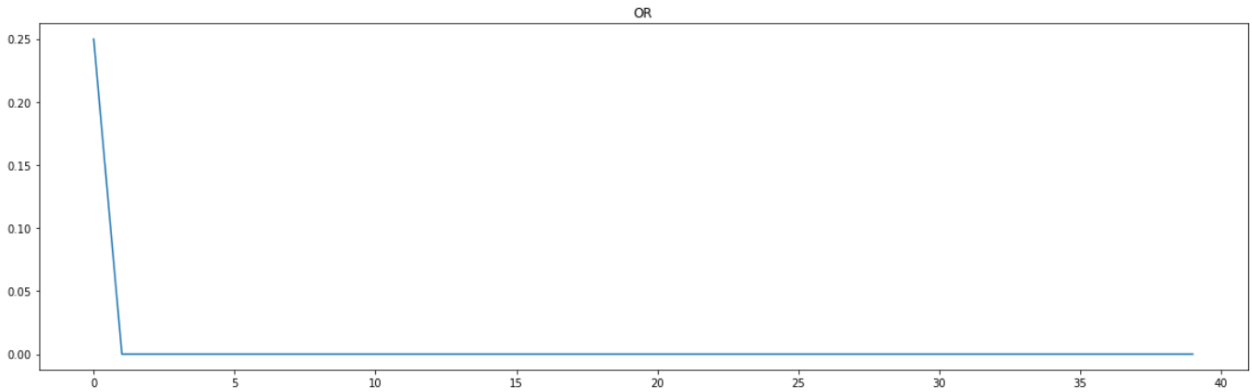
خطای شبکه(در گزارش ۱۸ تکرار از خطای شبکه قرار داده شده است و عملا از تکرار ۱۰ به بعد شبکه برای دو گیت AND و OR آموزش میبینند):

مشاهده منحنی خطای شبکه در هر مرحله برای هر دروازه منطقی:

دروازه AND: که مشاهده میشود پس از چندین تکرار یادگیری درومش انجام شده و خطای شبکه به صفر میرسد



دروازه OR: که مشاهده میشود پس از چندین تکرار یادگیری درومش انجام شده و خطای شبکه به صفر میرسد



و درنهایت دروازه XOR که مشهاده میشود داری خطای ثابت 0.5 است و یادگیری بعد از هر تعداد تکرار نیز رویش انجام نمیشود. علت این اتفاق این است که شبکه طراحی شده یک لایه است و برای تعلیم شبکه XOR حداقل به ۲ لایه احتیاج داریم و برای همین یادگیری در سطحی پایینتر از سطح موردنیاز برای تعلیم XORانجام میشود و در این شبکه هیچگاه XOR تعلسیم نمیبیند.