به نام خدا

پروژه دوم درس هوش محاسباتی

شبیه سازی یادگیری و تشخیص الگو ها در مغز انسان با یک حافظه ی خطی

نام دانشجو:

امیرحسین شریفی‌صدر ۹۷۳۳۰۴۴

استاد: دکتر سیدصالحی

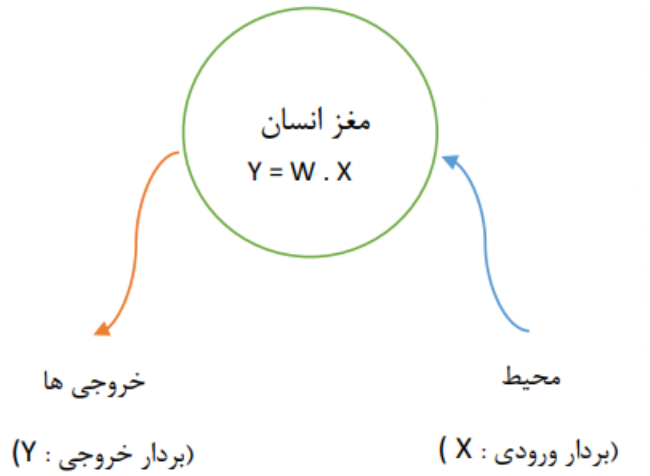
تدریسیار: خانم خلیلی

پاییز ۱۴۰۰

مقدمه :

یادگیری در مغز به شیوه‌ای است که در هر بار ورودی‌هایی را از محیط دریافت میکند و به مرور آن‌ها را با الگویی یاد میگیرد و خروجی‌شان را تولید میکند.

که میتوان آن‌را به شکل مقابل نشان داد:



حال به شبیه سازی این برنامه میپردازیم.

شبیه‌سازی با زبان برنامه‌نویسی پایتون و در jupyter lab انجام شده است.

کتاب‌خانه‌های مورد استفاده :

```
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

در ابتدا ورودی‌های خود را به صورت تصادفی تولید میکنیم.

بنده از ۱۰۰ ورودی استفاده کرده‌ام که بین ۰ تا ۲۰ قرار دارند. هر ورودی یک نقطه در فضا است که دارای سه بعد x,y,z است. این ورودی ها طبق گفته سوال باید در یک صفحه انتخاب شوند که ورودی‌های تولید شده توسط بنده همگی در صفحه x=z قرار دارند.

در زیر کد مربوط به آن دیده میشود:

```
x = np.random.randint(0, 20, size=(101, 3))
x[0:100, 0] = x[0:100, 2]
print(x)
```

همانطور که مشخص است در خط ابتدایی ۱۰۱ نقطه به صورت تصادفی تولید میشوند(نقطه ۱۰۱ در آخر دوباره توسط خودمان به عنوان test مقدار گذاری میشود.)

و در خط دوم این نقاط به به صفحه x=z انتقال میابند.

تعدادی از نقاط تولید شده را در زیر روبه‌رو میکنید:

```
[[ 6 15  6]
 [14  0 14]
 [17  9 17]
 [ 4  2  4]
 [ 8 19  8]
 [19 13 19]
 [10 12 10]
 [16  0 16]
 [10 13 10]
 [ 8 17  8]
 [ 6 19  6]
 [ 1  4  1]
```

در این مرحله همانطور که صورت سوال از ما خواسته است وزن یادگیری اولیه که هیچ قدرت یادگیری‌ای ندارد را تولید میکنیم که از مرتبه صفر است یعنی تمام درایه‌های آن تقریبا نزدیک صفر هستند.:

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

مقدار w را در زیر مشاهده میکنیم:

```
[[0.00030596 0.00060049 0.00043281]
 [0.00055524 0.00074738 0.00069169]
 [0.00084396 0.00092415 0.00086568]]
```

هدف از این برنامه آموختن و آپدیت کردن w به شیوه‌ایست که خروجی‌هایمان به مقادیر ورودی کاملا نزدیک شوند و در آخر عمل یادگیری صورت بگیرد. که این آموزش و یادگیری با اعمال الگوریتم LMS (least mean square) صورت میگیرد.

اصلاح وزن : $W(new) = W(old) + n X^T * (D - Y)$

بنده در از بین ۱۰۰ ورودی‌ای که به شکل تصادفی تولید شده بودند از ۸۰تای آنها برای train کردن استفاده کردم و بعد از تولید w که یادگیری روی آن صورت گرفته است ۲۰ ورودی دیگر را با آن w تست کردم و مشاهده شد که w به درستی آموزش دیده است و خروجی‌های موردانتظار تولید شدند.

مرحله train کردن داده‌ها و آپدیت w :

```
for i in range (0,80):
    y[i:i+1 , 0:3] = np.dot(x[i:i+1, 0:3],w)
    e[i:i+1 , 0:3] = (x[i:i+1 , 0:3] - y[i:i+1 , 0:3])
    e_mean[i:i+1 , 0:1] = np.mean(e[i:i+1 , 0:3])
    x1 = np.transpose(x[i:i+1 , 0:3])
    w = w + 0.001*(np.dot(x1,e[i:i+1, 0:3]))
```

y خروجی ما، e ارور ما در هر مرحله از train کردن و e_mean میانگین مختصات خطاها در هر مرحله است که یعنی از ۳ بعد آن میانگین گرفته شده و در یک نقطه ذخیره شده است تا بتوانیم آن را در نمودار دوبعدی نشان دهیم.

در مرحله اول با استفاده از ضرب داخلی w و ورودی اولمان یک خروجی تولید میکنیم.

سپس اختلاف این خروجی با مقدار مطلوب خروجی که باید برابر با ورودی باشد را به دست آورده که همان ارور است. حال برای آپدیت کردن w از فرمول گفته شده اسفاده میکنیم که در آن احتیاج است ترانهاده x محاسبه شود.

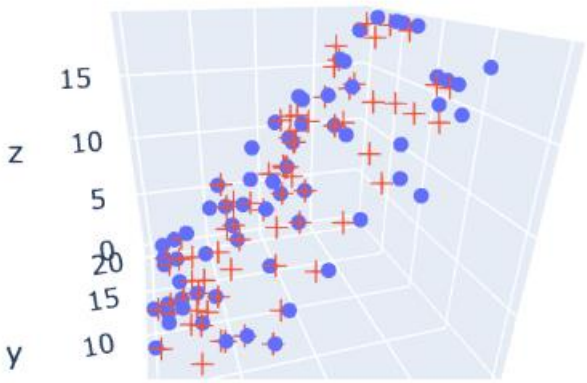
در روبه‌رو یک مثال از ورودی و ترانهاده آن را مشاهه میکنیم:

```
[[2 5 2]]
[[2]
 [5]
 [2]]
```

حال پس از انجام تمام این مراحل به تعداد داده‌های trainمان به w نهایی که یادگیری رو آن صورت گرفته است میرسیم:

```
[[0.49216944 0.00759833 0.49190263]
 [0.02015444 0.98118013 0.02015882]
 [0.49161304 0.00754531 0.49187633]]
```

نتیایج خروجی بعد از ۸۰ بار train:



دایره‌های آبی رنگ ورودی هستند

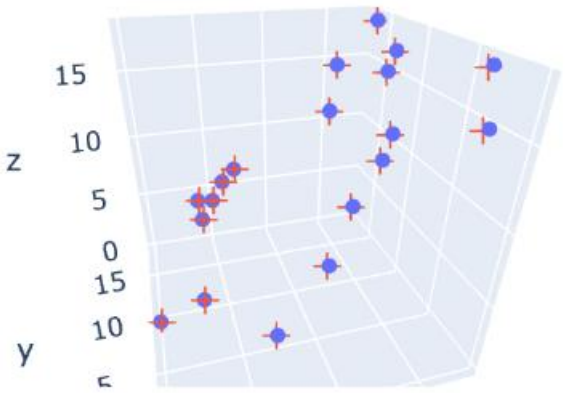
ضربدرهای قرمز رنگ مقادیر خروجی هستند

مشاهده میشود که مرور خروجی‌ها دقیق‌تر شده و به ورودی نزدیک تر میشوند.

سپس ۲۰ داده به عنوان داده تست(test) به عنوان ورودی میدهیم و با استفاده از w بالا که بعد از ۸۰ بار train تولید شده است خروجی را برای این ۲۰ نقطه تست به دست میآوریم.

```
for i in range (80,100):
    y[i:i+1 , 0:3] = np.dot(x[i:i+1, 0:3],w)
    e[i:i+1 , 0:3] = (x[i:i+1 , 0:3] - y[i:i+1 , 0:3])
```

خروجی‌های این ۲۰ داده تست را در زیر مشاهده میکنیم:



دایره‌های آبی رنگ ورودی هستند

ضربدرهای قرمز رنگ مقادیر خروجی هستند

در اینجا مشاهده میشود که شبکه کاملا آموزش دیده‌است و هر خروجی تقریبا با مقدار مورد انتظار برابری میکند.

البته باید حواسمان باشد که در بخش trainکردن، خروجی‌ها کاملا روی ورودی‌ها fit نشوند چون در این حالت شبکه تنها نسبتا به همان ورودی‌های خاص پاسخ درست میدهد.

میزان ۸۰ داده train و ۲۰ داده تست میتوان گفت که مقایدر مناسبی هستند تا هم شبکه آموزش را به خوبی انجام دهد و هم دچارoverfit نشود.

حال برای تست یک نقطه دیگر نیز خارج از این صفحه‌ای که شبکه روی آن آموزش دیده است انتخاب میکنیم. چون ما میدانیم شبکه ما تنها برای ورودی‌هایی که در صفحه x=z هستند آموزش دیده است پس انتظار داریم که برای داده‌ای خارج از این صفحه **نتواند** خروجی مدنظر که همان داده است را تولید کند.

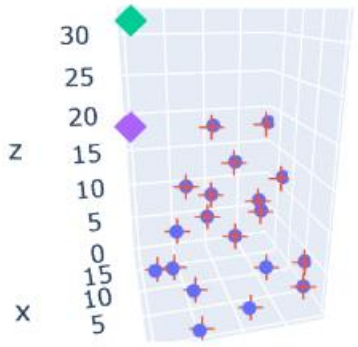
داده تست ما :

```
x[100:101 , 0:3] = [5,20,32]
```

که همانطور که در ابتدای گزارش گفته شد آنرا ورودی شماره‌ی ۱۰۱ام مان را قرار دادیم.

و خارج از صفحه x=z است.

خروجیش را در زیر میبینیم:



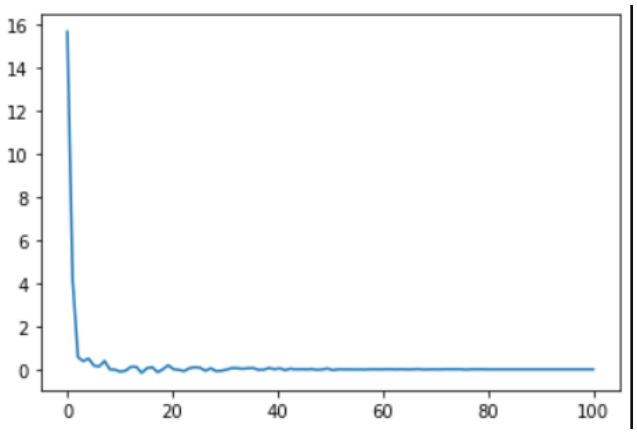
مربع سبز رنگ ورودی هستند

مربع بنفش رنگ مقدار خروجی است.

این دو نقطه روی همان نمودار نقاط تست درون صفحه ما رسم شدند.

میبینیم که نقطه خارج از صفحه دارای خروجی‌ای است که روی صفحه x=z که روی آن آموزش انجام شده است تصویر شده.

در آخر نمودار خطای میانگین که برای هر نقطه که داخل آرایه‌ی e_mean ریخته شده بود را اسم میکنیم که میبینم هر چه شبکه بیشتر آموزش میبیند و وزن یادگیری ما آپدیت میشود خطا بسیار کمتر میشود:



کد مربوط به نمایش نمودارها :

```
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter3d(x=x[0:80 , 0], y=x[0:80 , 1], z=x[0:80 , 2], mode='markers',   marker=dict(symbol="circle", size=4)))
fig.add_trace(go.Scatter3d(x=y[0:80 , 0], y=y[0:80 , 1], z=y[0:80 , 2], mode='markers',   marker=dict(symbol="cross", size=5)))

fig.show()

fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter3d(x=x[80:100 , 0], y=x[80:100 , 1], z=x[80:100 , 2], mode='markers',   marker=dict(symbol="circle", size=4)))
fig.add_trace(go.Scatter3d(x=y[80:100 , 0], y=y[80:100 , 1], z=y[80:100 , 2], mode='markers',   marker=dict(symbol="cross", size=6)))

x[100:101 , 0:3] = [5,20,32]
print(x[100:101 , 0:3])
y[100:101 , 0:3] = np.dot(x[100:101 , 0:3],w)
fig.add_trace(go.Scatter3d(x=x[100:101 , 0], y=x[100:101 , 1], z=x[100:101 , 2], mode='markers',   marker=dict(symbol="diamond", size=6)))
fig.add_trace(go.Scatter3d(x=y[100:101 , 0], y=y[100:101 , 1], z=y[100:101 , 2], mode='markers',   marker=dict(symbol="diamond", size=6)))

fig.show()


t = np.linspace(0,100,100)
plt.plot(t,e_mean)
plt.show()
```

به طور کلی نتیجه‌ای که میتوان از این پروژه گرفت این است که مغز ما به مرور هر داده‌ای که از محیطش دریافت کند را سعی میکند که یاد بگیرد و حال اگر داده‌ای مرتبط با داده‌های قبلی که آنهارا یاد گرفته است دریافت کند آن داده را نیز میتواند آموزش ببیند و خروجی مناسبی برایش تولید کند و لی اگر داده‌ای خرج از مباحث یادگیری به او داده شود سعی میکند که با توجه به فهم قبلی خود خروجی آن داده‌را روی چیزی که میداند تصویر کند. به قولی هر کس برداشت خودش از موضوعی خاص که در موردش اطلاعاتی ندارد را با فهم خودش انجام میدهد.