

به نام خدا

۹۸۳۱۲۶

امیررضا رجبی

گزارش پیاده سازی شبکه عصبی

قدم دوم و سوم :

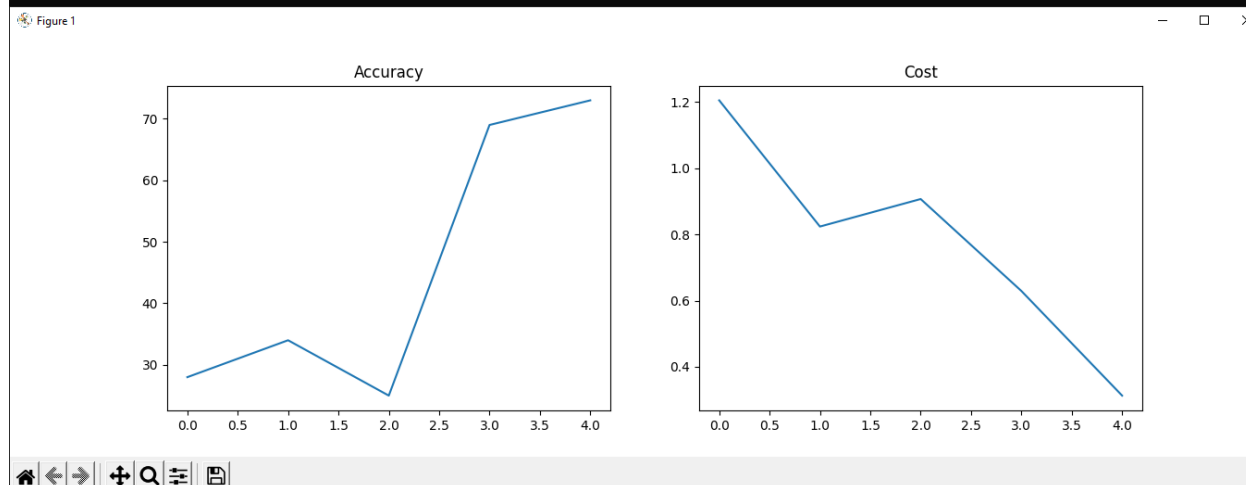
در قدم دوم قرار است شبکه ای که وزن دهی آن به صورت تصادفی است را آزمایش کرده و به طور میانگین باید دقت آن چیزی در حدود ۲۵ درصد باشد

در قدم سوم ما قرار است شبکه خود را با ۲۰۰ داده ای که در قدم دوم از آن استفاده کردیم آموزش دهیم اما با یک شرط آنکه بدون استفاده از وکتور کردن باشد

من کد این دو قسمت را یکی کرده و باهم اجرا میکنم.

ANN-NV.py

```
(AI-T) E:\CI-P\Fruit\ANN_Project>python ANN-NV.py
Accuracy: 25.3 % in 100 times for first time
#####
Time: 260435.93621253967 ms
Accuracy: 73.000 %
```



صد بار روی آن ۲۰۰ داده با وزنای مختلف بدون لرن (قدم ۲) پردیکت کرده که دقت میانگین آن ۲۵/۳ درصد درآمد

روند کاست نزولی است و روند دقت صعودی است و به ۷۳ درصد دقت آن رسید

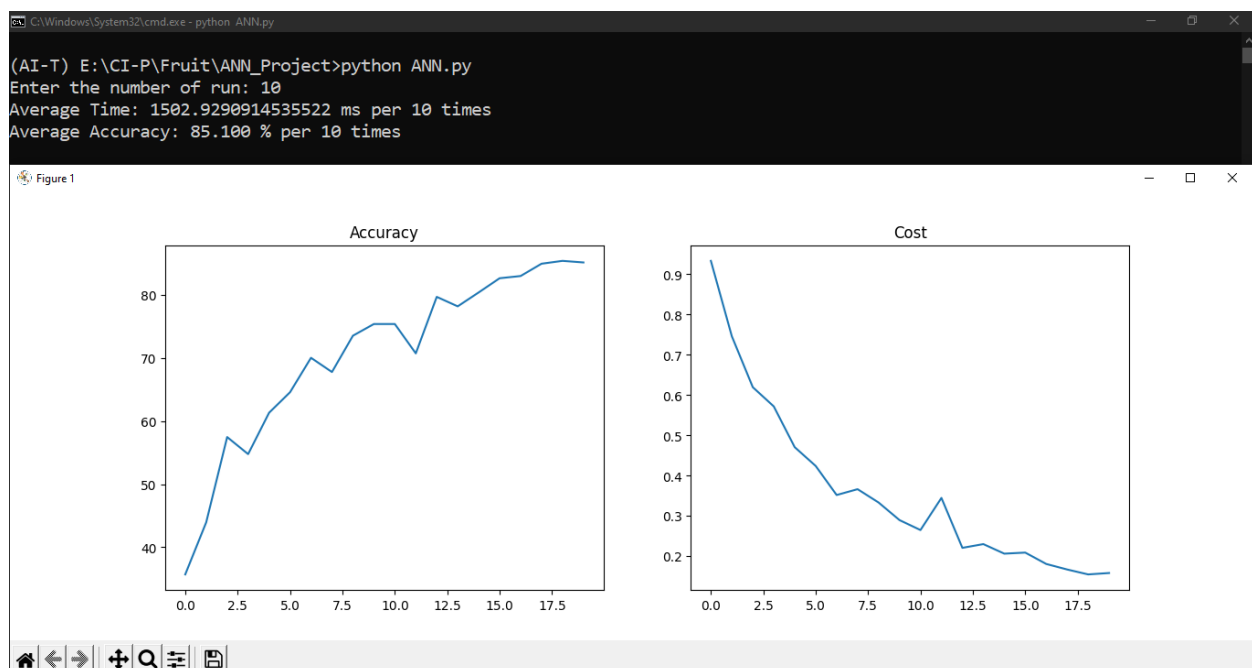
مدت زمان آن ۲۶۰ ثانیه است

قدم چهارم :

ANN.py

در این مرحله ما باید با وکتورسازی زمان کد را بهبود بدهیم
پس کارهایی که باید انجام دهیم :

وکتور کردن و افزایش دادن ایپاک به ۲۰ است ما کد رو ۱۰
بر ران گرفته و میانگین رو میگیریم



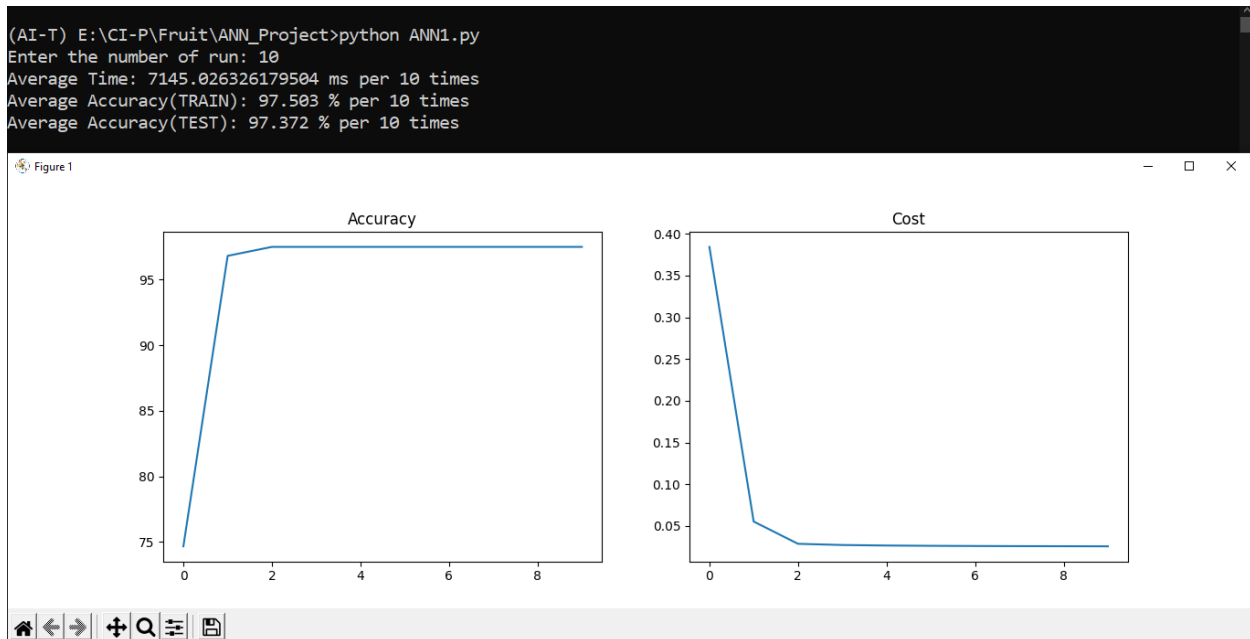
میانگین زمان اجرا به ۱/۵ ثانیه و دقت به ۸۵/۱ درصد
بهبود یافت

از کاست و دقت برای هر ایپاک در هر ده ران میانگین
گرفته وبعد پلات رو رسم کردم

قدم پنجم :

مراحل رو اجرا کرده و ده بار مانند قبل ران میگیریم

ANN1.py



به طور میانگین زمان اجرای آن ۷ ثانیه شد

و به طور میانگین دقت مجموعه ترین آن ۹۷/۵۰۳ و در
مجموعه تست ۹۷/۳۷۲ درصد شد

امتیازی اول :

من برای ۱۲۵ حالت از ترکیب ایپاک و بچ سایز و لرنینگ ریت بدست آوردم

که ۱۲۵ حالت تمامی تاپل های ممکن سه تایی از مقادیر پایین است که نتیجه آن در فایل زیر ذخیره شده است

Result_extra1.txt , ANN_Extra1.py

[illegible]

	5	10	15	20	50
Epoch	6	9	8	11	17

	0.01	0.03	0.1	0.3	1
L-R	3	5	9	14	20

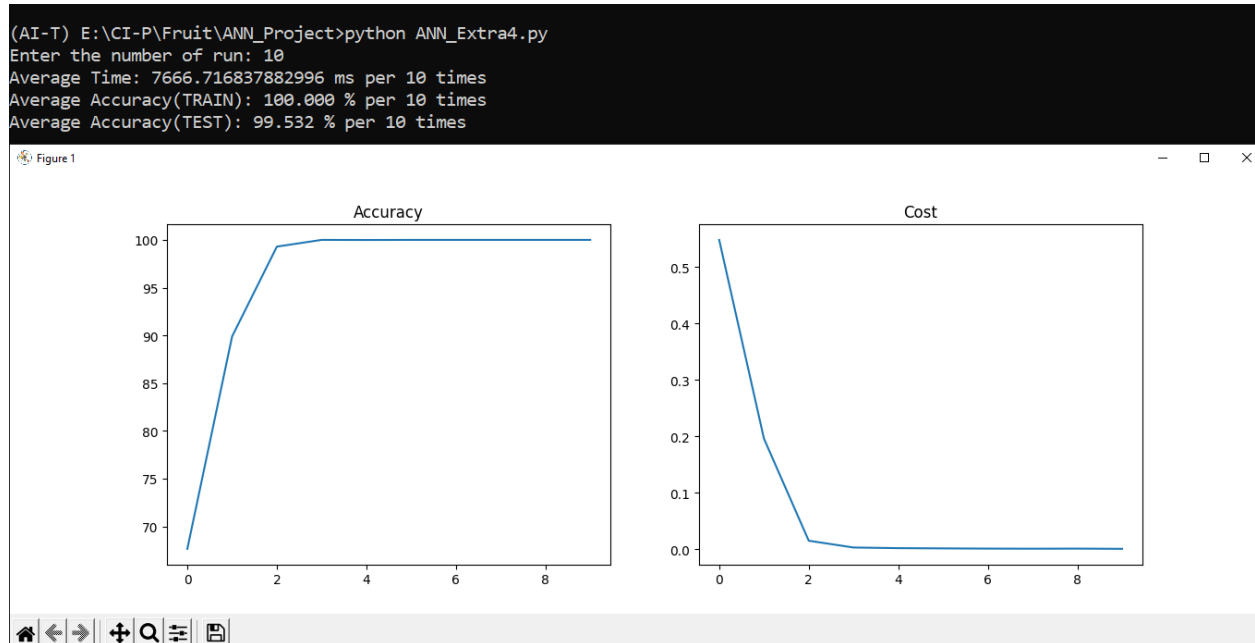
	10	20	30	50	100
B-Size	17	12	10	7	5

در جدول بالا آنهایی که دقت تست بالای ۹۸ را داشتند
مقدارشان آورده شده

لرنینگ ریت بین ۰/۳ و ۱ به سمت ۱ بسیار مناسب است
برای ایپاک مقدار ۵۰ مناسب است اما زمان را باید در نظر
گرفت که در این پروژه چون خیلی تغییری نمیکند پس مناسب
است اما باید اورفیت را در نظر گرفت
برای بچ سائز هم مقدار ۱۰ مناسب مناسب است

امتیازی چهارم :

برای لایه آخر از تابع سافتمکس استفاده میکنیم و برای مقایسه مانند قدم پنجم عمل میکنیم



زمان یک مقدار بالاتر میرود چون تابع سافتمکس هزینه بیشتری دارد . میانگین ترین ۱۰۰ درصد در ده ران و میانگین تست ۹۹/۵۳۲ است که جفتشون از قدم پنجم بهتر هستند . نتیجه که گرفتیم بهبود پیدا کرده به این علت که سافتمکس می آید اون اختلاف رو بیشتر میکند یعنی اونی که بیشتر هست رو به ۱ نزدیک و بقیه رو به صفر نزدیک میکند

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

امتیازی سوم :

من دو میوه نارگیل و گیلان از این دیتاست رو اضافه کردم

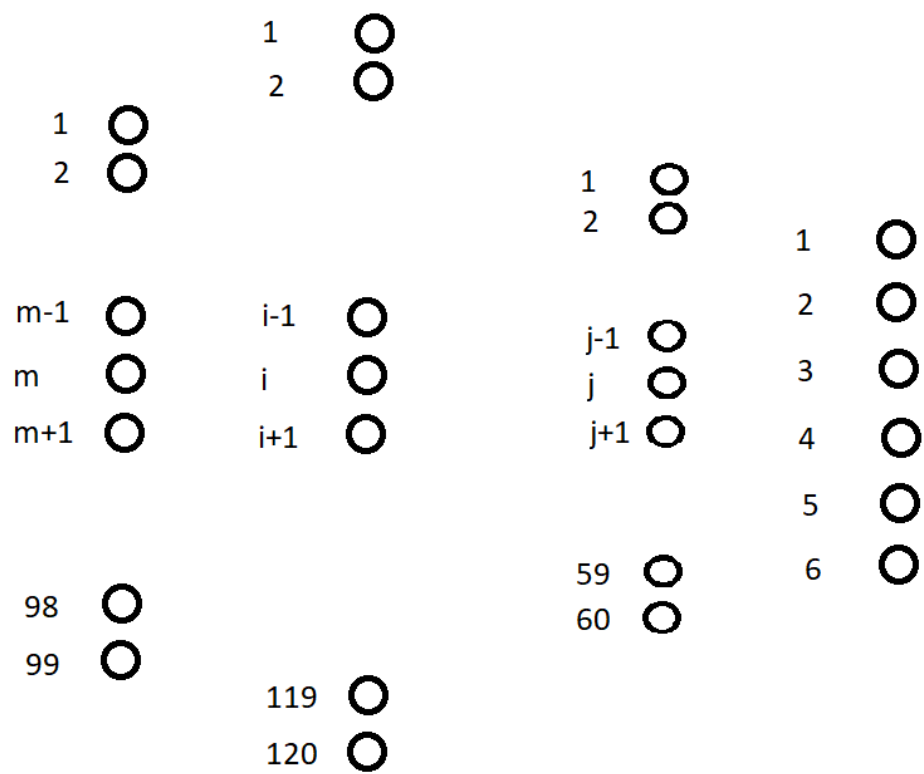
```
C:\Windows\System32\cmd.exe

(AI-T) E:\CI-P\Fruit\EXTERA3>python ANN1.py
Case 0:
Time: 30024.433135986328 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 99.294 %
#####
Case 1:
Time: 30016.329765319824 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 98.992 %
#####
Case 2:
Time: 29220.452547073364 ms
Accuracy(TRAIN): 83.356 %
Accuracy(TEST): 82.560 %
#####
Case 3:
Time: 30077.3184299469 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 99.597 %
#####
Case 4:
Time: 29599.395275115967 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 98.992 %
#####
Case 5:
Time: 29519.404649734497 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 99.194 %
#####
Case 6:
Time: 29789.364099502563 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 99.194 %
#####
Case 7:
Time: 29464.415311813354 ms
Accuracy(TRAIN): 100.000 %
Accuracy(TEST): 98.387 %
#####

(AI-T) E:\CI-P\Fruit\EXTERA3>
```

Learning_rate = 1, batch_size = 30, epoch = 30

مقادیر بالا و تعداد لایه ها و تعداد نوروں ها در هر لایه با سعی و خطا بدست آمده اند



امتیازی دوم :

Momentum:

$$W_{new} = W_{old} + \Delta W_{new}$$

$$\Delta W_{new} = \alpha \Delta W_{old} + \eta dW_{new}$$

در واقع داریم برای آپدیت این لایه از وزن لایه های از تغییرات لایه پیشین استفاده میکنیم و این روند رو تسریع میکند و جلوگیری میکند از توقف در مینیمم محلی که آلفا مقدار ممنتوم اس که عددی بین ۰ و ۱ است

```
C:\Windows\System32\cmd.exe - python ANN_momentum.py
(AI-T) E:\CI-P\Fruit\ANN_Project>python ANN_momentum.py
Enter the number of run: 10
Average Time: 9787.67466545105 ms per 10 times
Average Accuracy(TRAIN): 87.503 % per 10 times
Average Accuracy(TEST): 87.372 % per 10 times
Average Time: 9787.67466545105 ms per 10 times
Average Accuracy(TRAIN): 57.513 % per 10 times
Average Accuracy(TEST): 57.462 % per 10 times
```

Figure 1

