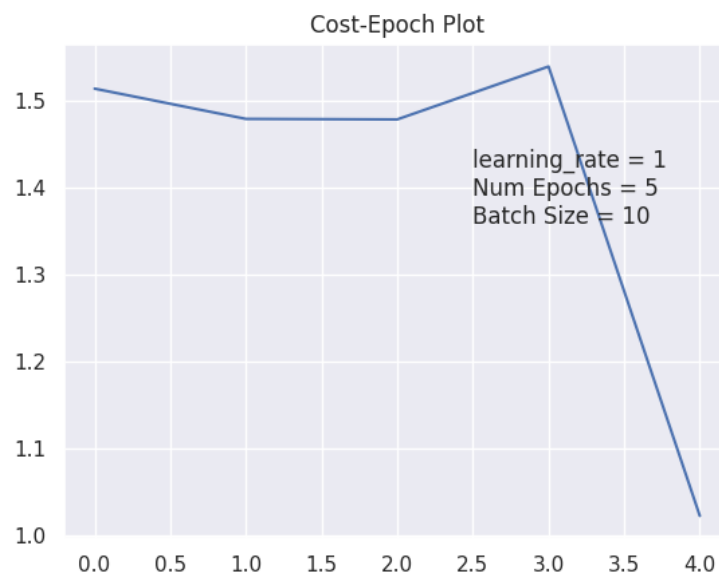


۱. گام دوم، Feedforward

```
divar@divar-ThinkPad-E14: ~/PycharmProjects/ANN | 111x17 | pts/1  
(venv) └─(~/PycharmProjects/ANN)───(divar@divar-ThinkPad-E14:pt  
s/1)└─  
└─(23:08:25)─> python3 feedforward.py  
Accuracy: 0.25076452599388377  
(venv) └─(~/PycharmProjects/ANN)───(divar@divar-ThinkPad-E14:pt  
s/1)└─  
└─(23:08:27)─> █
```

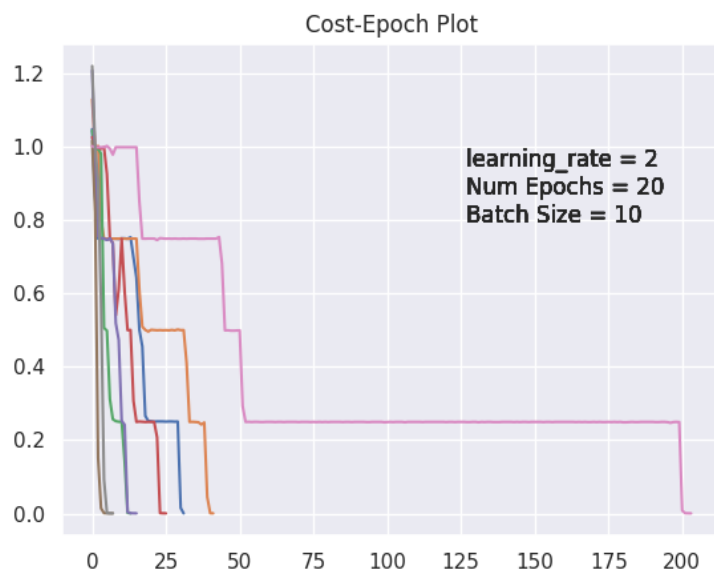
۲. گام سوم، پیاده‌سازی Backpropagation

```
(venv) ~/PycharmProjects/ANN | 146x38 | pts/1  
s/1) → python3 backpropagation.py  
EPOCH#0 | 20/20 [00:56<00:00, 2.81s/it]  
EPOCH#1 | 20/20 [01:01<00:00, 3.07s/it]  
EPOCH#2 | 20/20 [00:59<00:00, 2.97s/it]  
EPOCH#3 | 20/20 [00:57<00:00, 2.90s/it]  
EPOCH#4 | 20/20 [00:57<00:00, 2.89s/it]  
Accuracy: 0.27  
  
/home/divar/PycharmProjects/ANN/utils.py:49: UserWarning: Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure  
.plt.show()  
Execution Time: 292.7627909679941s  
(venv) ~/PycharmProjects/ANN | 146x38 | pts/1  
s/1) →
```



3. گام چهارم، پیاده‌سازی Vectorization

نمودار هزینه‌ها:



میانگین دقت و میانگین زمان اجرای الگوریتم، طی ۱۰ بار اجرا:


```
Activities Terminal 23:46 9 دسامبر • divar@divar-ThinkPad-E14: ~/PycharmProjects/ANN | 146x38 | pts/1
(venv) [~/PycharmProjects/ANN] (divar@divar-ThinkPad-E14:pts/1)
└─(23:42:07)─> python3 test_model.py ─┬─(جزمعه آخفا مي, دسا مير09)
Trial #0
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #1
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #2
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #3
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #4
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #5
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #6
Train Accuracy: 0.9989806320081549
Test Accuracy: 0.9984894259818731
Trial #7
Train Accuracy: 0.9989806320081549
Test Accuracy: 0.9969788519637462
Trial #8
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
Trial #9
Train Accuracy: 0.9989806320081549
Test Accuracy: 1.0
AVG Train Accuracy: 0.9989806320081549
AVG Test Accuracy: 0.9995468277945619
(venv) [~/PycharmProjects/ANN] (divar@divar-ThinkPad-E14:pts/1)
└─(23:46:31)─> [ ] ─┬─(جزمعه آخفا مي, دسا مير09)
```

امتیازی‌ها:

۱. انتخاب یک ضریب یادگیری مناسب

برای این کار، مقادیر مختلف ضریب یادگیری را بر حسب تعداد Epochهای مورد نیاز برای همگرایی مقایسه کرده، و بهترین را انتخاب می‌کنیم. تعریفمان از همگرایی، زمانی است که میزان cost کمتر از یک میزان مشخص (در این جا 0.01) شود.

روند بالا در کد مازول find_lr پیاده شده است. در این مازول، به ازای هر ضریب یادگیری، میانگین تعداد ایپاک‌های مورد نیاز برای رسیدن به همگرایی (تعریف شده به عنوان convergence_cost) گزارش می‌شود.

تغییر تعداد Epochها:

Epochs	AVG Accuracy on Test Set
1	0.61
5	0.84
10	0.97

Learning Rate = 1, Batch Size = 10

تغییر اندازه‌ی Batch:

Batch Size	AVG Accuracy on Test Set
1	0.9734138972809667
5	0.924773413897281
10	0.8749244712990937

Epochs = 10, Learning Rate = 1

تغییر Learning Rate:

Learning Rate	AVG Accuracy on Test Set
0.5	0.9728096676737159
1	0.9235649546827794
2	0.35861027190332323

Epochs = 10, Batch Size = 10

۲. روش‌های پیشرفته‌تر در SGD

فارغ از این که برای ضرب یادگیری چه مقداری تعیین کنیم، در مرحله‌ای از اجرای الگوریتم آن مقدار ممکن است بیش از حد بزرگ یا کوچک باشد. در چنین شرایطی نیاز به یک ضرب یادگیری متغیر خواهیم داشت، که برای آن چندین الگوریتم مانند Adam, RMSProp, Momentum پیشنهاد شده‌اند.

روش Momentum در پروژه پیاده شده است. این روش به این صورت عمل می‌کند که در هر مرحله، علاوه بر حاصل ضرب گرادیان در ضرب یادگیری، یک جمله جدید هم روی تغییرات وزن متأثر است. این جمله، حاصل ضرب یک ضرب جدید (ضرب Momentum) در گرادیان میزان تغییر وزن در پیمایش قبلی است.

Backpropagation Training

$$w_{i,j}(t+1) = w_{i,j}(t) + \eta \Delta w_{i,j} + \alpha \Delta w_{i,j}(t)$$

- where:
 - delta is the weight change
 - eta is the learning rate
 - alpha is the momentum term

به این ترتیب، نحوه‌ی به‌روزرسانی وزن‌ها به شکل زیر می‌شود.

```
w1_velocity = learning_rate * (d_w1 / batch_size) + momentum * w1_velocity
w1 -= w1_velocity

w2_velocity = learning_rate * (d_w2 / batch_size) + momentum * w1_velocity
w2 -= w2_velocity

w3_velocity = learning_rate * (d_w3 / batch_size) + momentum * w3_velocity
w3 -= w3_velocity

b1 -= learning_rate * (d_b1 / batch_size)
b2 -= learning_rate * (d_b2 / batch_size)
b3 -= learning_rate * (d_b3 / batch_size)
```

با اجرای کد فایل mometum به نتایج زیر می‌رسیم.

```
divar@divar-ThinkPad-E14:~/PycharmProjects/ANN | 146x38 | pts/1
EPOCH#11
EPOCH#12
EPOCH#13
EPOCH#14
EPOCH#15
EPOCH#16
EPOCH#17
EPOCH#18
EPOCH#19
Accuracy: 1.0

EPOCH#0
EPOCH#1
EPOCH#2
EPOCH#3
EPOCH#4
EPOCH#5
EPOCH#6
EPOCH#7
EPOCH#8
EPOCH#9
EPOCH#10
EPOCH#11
EPOCH#12
EPOCH#13
EPOCH#14
EPOCH#15
EPOCH#16
EPOCH#17
EPOCH#18
EPOCH#19
Accuracy: 0.76

AVG Accuracy: 0.8450000000000001
AVG Execution Time: 2.059368713699223
(venv) └─(~/PycharmProjects/ANN)───(divar@divar-ThinkPad-E14:pt
s/1)─┐
└─(01:33:01)─┐ ──(جمعه، دسامبر 10)─┐
```

۳. تست برنامه برای کلاس‌های پیش‌تر

دو کلاس 1 Cherry و Pineapple به کلاس‌های دیتاست اضافه شده‌اند. برای این کار، نیاز به تغییر فایل‌های Load_Dataset.py و Feature_Extracion_*.py بود. فایل‌های جدید با ابعاد به‌روز شده ضمیمه شده‌اند.

نتیجه‌ی خروجی برنامه برای این حالت جدید به شکل زیر است.

```
(venv) ~/PycharmProjects/ANN (divar@divar-ThinkPad-E14:pt
s/1)
└─(02:38:56)─> python3 test_model.py
Trial #0 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #1 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #2 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #3 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #4 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #5 Train Accuracy: 0.7497451580020388 Test Accuracy: 0.7447129909365559
Trial #6 Train Accuracy: 0.9989806320081549 Test Accuracy: 0.9969788519637462
Trial #7 Train Accuracy: 0.7492354740061162 Test Accuracy: 0.7492447129909365
Trial #8 Train Accuracy: 0.9989806320081549 Test Accuracy: 1.0
Trial #9 Train Accuracy: 0.9989806320081549 Test Accuracy: 0.9984894259818731
AVG Train Accuracy: 0.9490825688073394
AVG Test Accuracy: 0.9489425981873112
(venv) ~/PycharmProjects/ANN (divar@divar-ThinkPad-E14:pt
s/1)
└─(02:40:37)─> 
```

۴. استفاده از Softmax

تابع Softmax در فایل utils پیاده شده، و از آن به عنوان تابع فعال‌سازی لایه‌ی آخر (محاسبه‌ی a_3) در ماژول softmax استفاده شده است. با اجرای برنامه در چنین حالتی خواهیم داشت:


```
(venv) └─(~/PycharmProjects/ANN)──────────────────────────────────┐divar@divar-ThinkPad-E14:pt  
├─(00:19:08)→ python3 softmax.py──────────────────────────────────┘  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.53it/s]  
Accuracy: 1.0  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.83it/s]  
Accuracy: 0.465  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.86it/s]  
Accuracy: 0.31  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.61it/s]  
Accuracy: 1.0  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.89it/s]  
Accuracy: 1.0  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 11.01it/s]  
Accuracy: 1.0  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 10.80it/s]  
Accuracy: 0.81  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:02<00:00, 8.31it/s]  
Accuracy: 0.645  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:02<00:00, 9.73it/s]  
Accuracy: 0.245  
  
100%|██████████████████████████████████████████████████████████| 20/20 [00:01<00:00, 10.41it/s]  
Accuracy: 0.98  
  
AVG Accuracy: 0.7455  
AVG Execution Time: 1.9877663978986675  
(venv) └─(~/PycharmProjects/ANN)──────────────────────────────────┐divar@divar-ThinkPad-E14:pt  
├─(00:19:29)→────────────────────────────────────────────────────────┘  
└─(جزعه، دسا میر 10)────────────────────────────────────────────────
```

همان‌طور که مشاهده می‌شود میزان دقت به 0.74 رسیده و میانگین زمان اجرای برنامه نیز به 1.98 ثانیه رسیده است.