

THESE DIAGRAMS ARE FOR PART d.

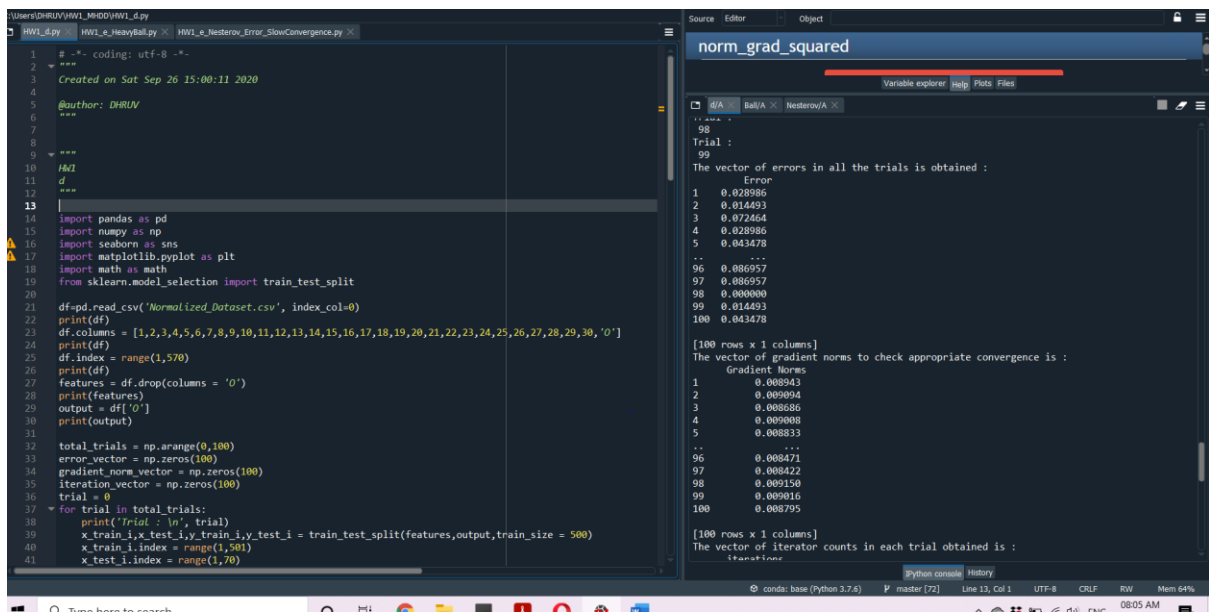
(It has a rather high run time)

Few Results for Gradient Descent

```
98
Trial :
99
The vector of errors in all the trials is obtained :
Error
1    0.028986
2    0.014493
3    0.072464
4    0.028986
5    0.043478
..    ...
96    0.086957
97    0.086957
98    0.000000
99    0.014493
100   0.043478

[100 rows x 1 columns]
The vector of gradient norms to check appropriate convergence is :
Gradient Norms
1    0.008943
2    0.009094
3    0.008686
4    0.009008
5    0.008833
..    ...
96    0.008471
97    0.008422
98    0.009150
99    0.009016
100   0.008795

[100 rows x 1 columns]
The vector of iterator counts in each trial obtained is :
iterations
```



```

[100 rows x 1 columns]
The vector of iterator counts in each trial obtained is :
iterations
1 5332.0
2 4651.0
3 5252.0
4 5153.0
5 4587.0
..
96 4171.0
97 5191.0
98 4763.0
99 4818.0
100 4387.0

[100 rows x 1 columns]
The average iterations till the accuracy of 10^-6 is satisfied in all 100 trials is :
4752.54
The average error obtained in all the 100 trials with 500 iterations in each trial is :
0.05260869565217392
The average error in percentage is :
5.2608695652173925
The average error in terms of average test data output mismatches is :
3.6300000000000001

```

In [2]: |

Few Results for Heavy Ball

```
50
Trial :
99
The vector of errors in all the trials is obtained :
      Error
1      0.014493
2      0.028986
3      0.086957
4      0.086957
5      0.000000
..      ...
96     0.043478
97     0.028986
98     0.072464
99     0.000000
100    0.057971

[100 rows x 1 columns]
The vector of gradient norms to check appropriate convergence is :
      Gradient Norms
1      0.009005
2      0.008738
3      0.008597
4      0.008686
5      0.009153
..      ...
96     0.008884
97     0.008880
98     0.008633
99     0.009110
100    0.008753

[100 rows x 1 columns]
The vector of iterator counts in each trial obtained is :
      iterations
```

```

[100 rows x 1 columns]
The vector of iterator counts in each trial obtained is :
      iterations
1          941.0
2          850.0
3          987.0
4          879.0
5          951.0
..          ...
96          856.0
97          881.0
98          934.0
99          990.0
100         995.0

[100 rows x 1 columns]
The average iterations till the accuracy of  $10^{-6}$  is satisfied in all 100 trials is :
  931.29
The average error obtained in all the 100 trials with 500 iterations in each trial is :
  0.0518840579710145
The average error in percentage is :
  5.18840579710145
The average error in terms of average test data output mismatches is :
  3.5800000000000005

In [2]: |

```

As can be seen Heavy Ball converges much faster than normal Gradient Descent

For Nesterov

```
Notice that grad_b_l and grad_w_l are the same for grad_w_b and grad_s_f

As already stated before these terms were independent of the vector at which gradient was evaluated
and stayed same for a given trial irrespective of iteration. Their calculations thus were done
before the while loop which iterates through different w_b and s_f estimates

When gradient evaluated at s_f instead of w_b these same terms appear again for every iteration
within a trial and are exactly same as those evaluated for w_b

The ratio vector changes however and we've calculated it again for s_f
The subsequent loop uses this ratio vector to calculate both the gradient terms of s_f corresponding
to s and f both

====

norm_grad = np.linalg.norm(grad_w_b)

norm_grad_squared = (norm_grad)*(norm_grad)

r=0
sum_function = 0
for r in N:
    sum_function = sum_function + ((-y_train[r])*(np.dot(w,x_train[r,:])+b)) + math.log(1 + math.exp((
function_value = sum_function + 0.5*lam*((np.linalg.norm(w))**2)

if norm_grad_squared <= ((10**(-6))*(1+abs(function_value))):
    break
else:
    if iterator_count == 1:
        w_b_old = w_b
        w_b = w_b - (u)*(grad_w_b)
        s_f = w_b + n*(w_b - w_b_old)
    else:
        w_b_l = s_f - (u)*(grad_s_f)
        w_b_old = w_b
        w_b = w_b_l
        s_f = w_b + (n)*(w_b - w_b_old)
```

norm_grad_squared

	1	2	3	...	28	29	30
1	0.003224	-0.007436	0.025730	...	0.000126	0.000142	0.000029
2	0.005074	-0.001197	0.032233	...	0.000056	-0.000012	0.000004
3	0.005582	0.001967	0.038162	...	0.000129	0.000071	0.000004
4	-0.006554	0.002640	-0.034837	...	0.000346	0.000905	0.000216
5	0.006491	-0.005213	0.045426	...	0.000050	-0.000057	-0.000008
...
565	0.005238	0.002185	0.035258	...	0.000075	-0.000059	-0.000009
566	0.005728	0.008550	0.037436	...	0.000046	-0.000031	-0.000017
567	0.007764	0.027602	0.051280	...	0.000085	-0.000214	-0.000018
568	0.005749	0.008918	0.042752	...	0.000134	0.000105	0.000036
569	-0.008193	0.006756	-0.056682	...	-0.000147	-0.000004	-0.000017

[569 rows x 31 columns]

	1	2	3	...	28	29	30
1	1	1	1	...	1	1	1
2	1	1	1	...	1	1	1
3	1	1	1	...	1	1	1
4	1	1	1	...	1	1	1
5	1	1	1	...	1	1	1
...
565	1	1	1	...	1	1	1
566	1	1	1	...	1	1	1
567	1	1	1	...	1	1	1
568	1	1	1	...	1	1	1
569	0	0	0	...	0	0	0

Name: 0, Length: 569, dtype: int64
Trial :
0

The run times for Nesterov Model were rather high and so in the few trials I could observe I obtained approximate iterations of ~1000

Again it was, due to large run times, not possible for me to tune the momentum term to ensure a quick convergence

CURVES

Plots have been submitted in the folder

Which model would I select?

I would select the Heavy Ball one as it showed the best results for me, despite the Nesterov method being the most preferred one. Due to run time issues I couldn't tune the parameter of momentum in Nesterov, which weakened its results