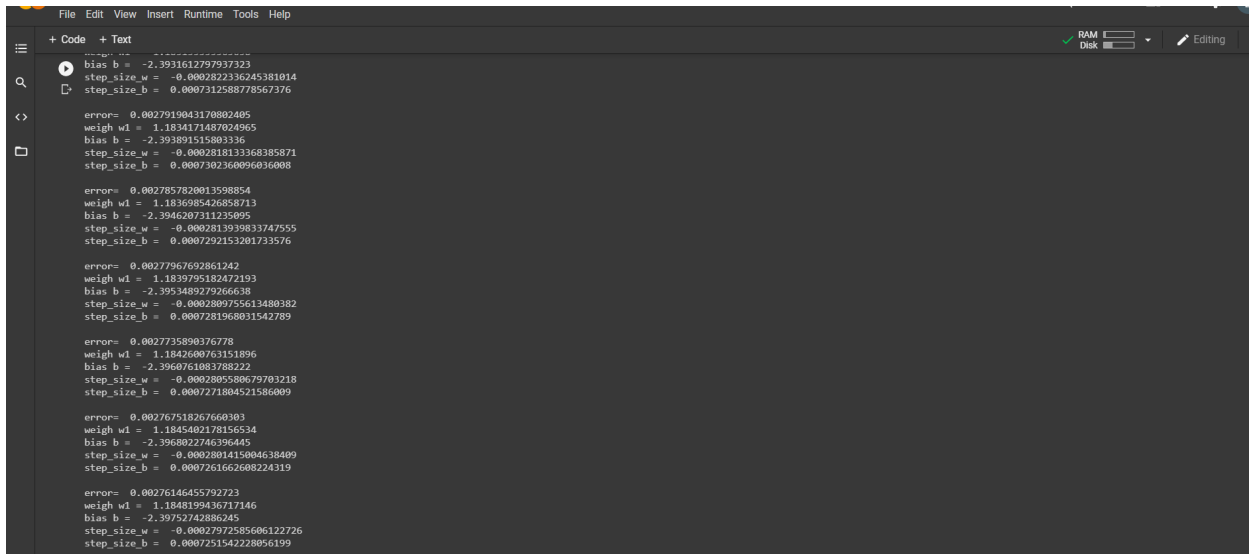


Comparison between Batch, Momentum-based, and Nesterov Accelerated Gradient Descent

1) Comparison between Batch and Momentum-based Gradient Descent

The following is the output that I got by running the uploaded script of Batch Gradient Descent(batch GD.py).



```
File Edit View Insert Runtime Tools Help
+ Code + Text
bias b = -2.3931612797937323
step_size_w = -0.0002822336245381014
step_size_b = 0.0007312588778567376

error= 0.0027919043170802405
weigh w1 = 1.1834171487024965
bias b = -2.393891515803336
step_size_w = -0.0002818133368385871
step_size_b = 0.0007302360096036008

error= 0.0027857820013598854
weigh w1 = 1.1836985426858713
bias b = -2.3946207311235095
step_size_w = -0.0002813939833747555
step_size_b = 0.0007292153201733576

error= 0.00277967692861242
weigh w1 = 1.1839795182472153
bias b = -2.3953489279266638
step_size_w = -0.0002809755613480382
step_size_b = 0.0007281968031542789

error= 0.0027735890376778
weigh w1 = 1.1842600763151896
bias b = -2.3960761083788222
step_size_w = -0.0002805580679703218
step_size_b = 0.0007271804521586009

error= 0.002767518267660303
weigh w1 = 1.1845402178156534
bias b = -2.3968022746396445
step_size_w = -0.00028014150804638409
step_size_b = 0.0007261662688224319

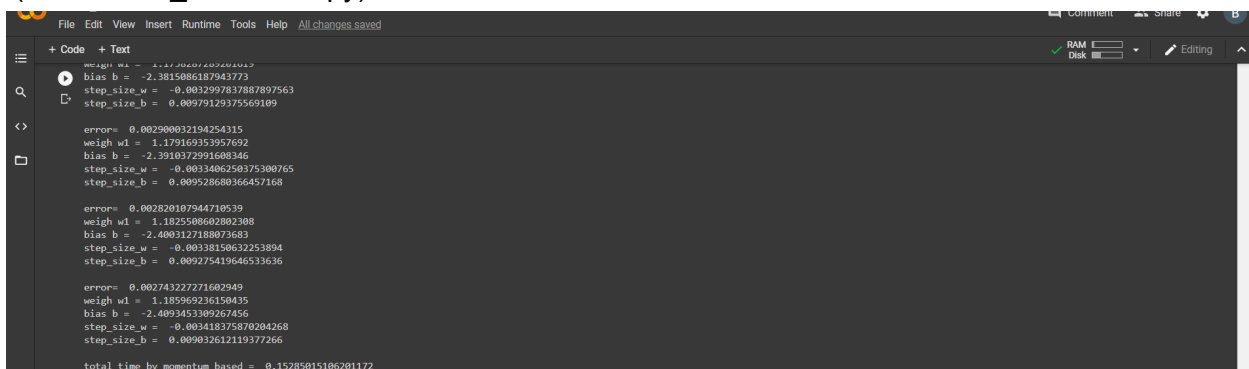
error= 0.00276146455792723
weigh w1 = 1.1848199436717146
bias b = -2.39752742086245
step_size_w = -0.00027972585606122726
step_size_b = 0.000725142228056199
```

Fig 1. Output of Batch Gradient descent for 999 iterations

Here, I have set:

- i) initial weight (w_1) = 0
- ii) bias (b) = 0
- iii) learning rate = 0.1
- iv) maximum epochs = 999

The following is the output that I got by running the uploaded script of Momentum-based Gradient Descent(momentum_based GD.py)



```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
weigh w1 = -2.377060016024541142
bias b = -2.3815086187943773
step_size_w = -0.0032997837887897563
step_size_b = 0.00979129375569109

error= 0.002900032194254315
weigh w1 = 1.179169353957692
bias b = -2.3910372991608346
step_size_w = -0.0033406250375300765
step_size_b = 0.009528680366457168

error= 0.002820107944710539
weigh w1 = 1.1825508602802308
bias b = -2.4003127188073683
step_size_w = -0.00338150632253894
step_size_b = 0.009275419646533636

error= 0.00274322771602949
weigh w1 = 1.185969236150435
bias b = -2.4093453309267456
step_size_w = -0.003418375870204268
step_size_b = 0.009032612119377266

total time by momentum based = 0.15285015106201172
```

Fig 2. Output of Momentum-based Gradient Descent

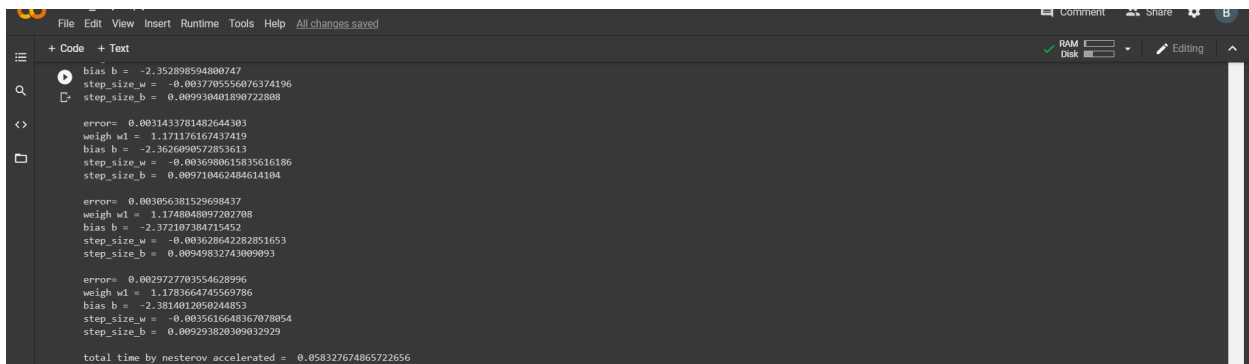
Here, I have set:

- i) initial weight (w_1) = 0
- ii) bias (b) = 0
- iii) learning rate = 0.1
- iv) maximum epochs = 84

It can be observed from the fig1 and 2 that for the same function with the same training set, initial weight (w_1), bias (b), and learning rate; Batch Gradient Descent gives an error of 0.002 after 999 iterations whereas Momentum-based Gradient Descent gives almost the same output only after 84 iterations. Therefore, it can be inferred that Momentum-based Gradient Descent (and thus Nesterov Accelerated Gradient Descent) is way efficient than Batch Gradient Descent.

2) Comparison between Momentum-based Gradient Descent and Nesterov Accelerated Gradient Descent

The following is the output that I got by running the uploaded script of Nesterov Accelerated Gradient Descent(nesterov_accelerated GD.py)



```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
bias b = -2.352898594880747
step_size_w = -0.0037785556076374196
step_size_b = 0.009930481890722808

error= 0.0031433781482644383
weigh w1 = 1.171176167437419
bias b = -2.3626990572853613
step_size_w = -0.0036988615835616186
step_size_b = 0.009710462484614104

error= 0.003056381529698437
weigh w1 = 1.1748048097202708
bias b = -2.372107384715452
step_size_w = -0.003628642282851653
step_size_b = 0.00949832743009093

error= 0.0029727703554628996
weigh w1 = 1.1783664745569786
bias b = -2.3814012050244853
step_size_w = -0.0035616648367078054
step_size_b = 0.009293820309032929

total time by nesterov accelerated = 0.058327674865722656
```

Fig 3. Output of Nesterov Accelerated Gradient Descent

Here, I have set:

- i) initial weight (w_1) = 0
- ii) bias (b) = 0
- iii) learning rate = 0.1
- iv) maximum epochs = 84

It can be noted from fig 2 and 3 that both Momentum-based Gradient Descent and Nesterov Accelerated Gradient Descent gives almost the output for the same function, training set, parameters, and hyper-parameters; but

the time taken by Nesterov Accelerated Gradient Descent to do this quite less than what is taken by Momentum-based Gradient Descent.

NOTE: This time taken can keep fluctuating. The above shown time are the ones which I got when I ran both the scripts for the first time. Also, here we have taken only 2 training data, but when dealing with real-world datasets where we consider lakhs of data inputs, during that instance, the total time taken by an algorithm will impact a major difference.

Moreover, below I have attached the different types of graphs for Momentum-based Gradient Descent and Nesterov Accelerated Gradient Descent.

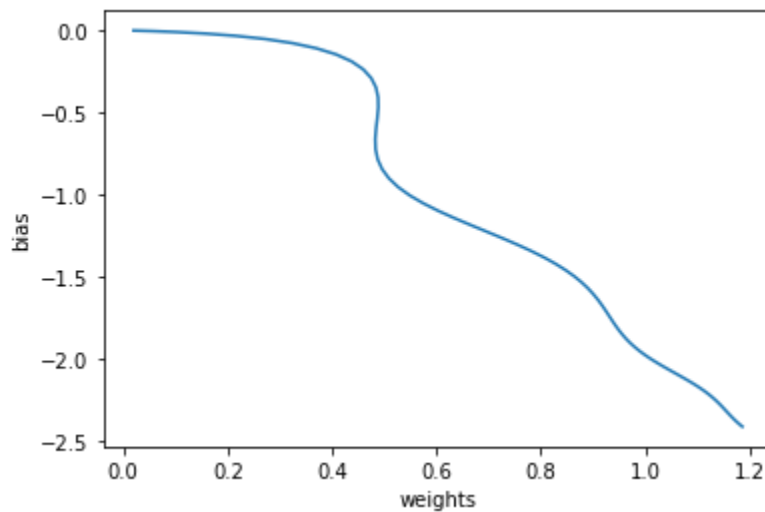


Fig 4. Weight vs Bias graph in Momentum-based Gradient Descent

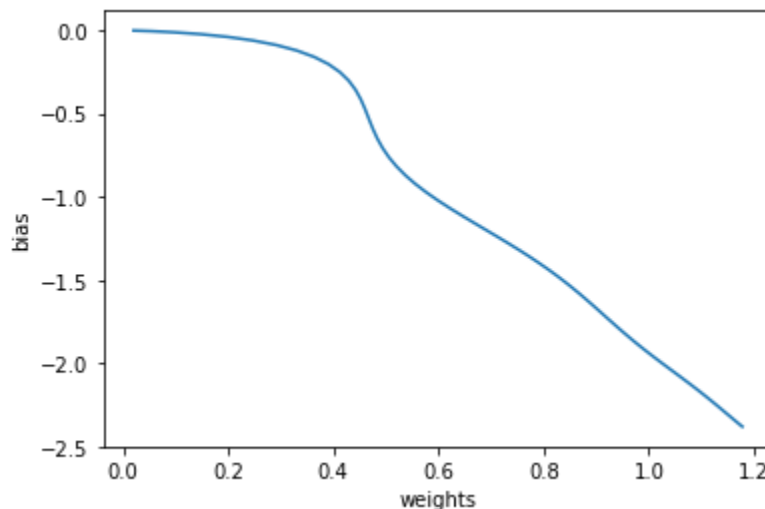


Fig 5. Weight vs Bias graph in Nesterov Accelerated Gradient Descent

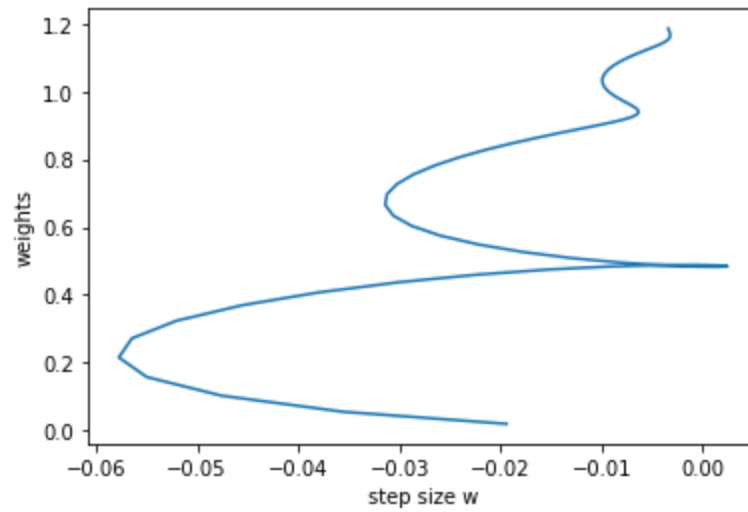


Fig 6. Step size for weights vs weights graph in Momentum-based Gradient Descent

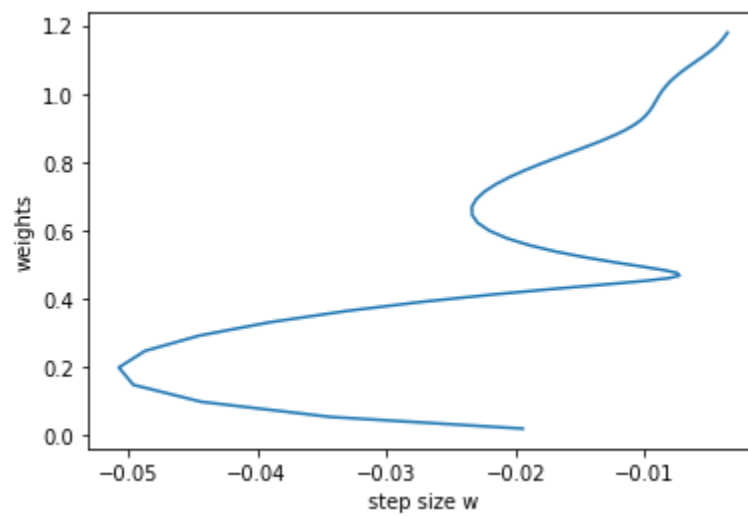


Fig 7. Step size for weights vs weights graph in Nesterov Accelerated Gradient Descent

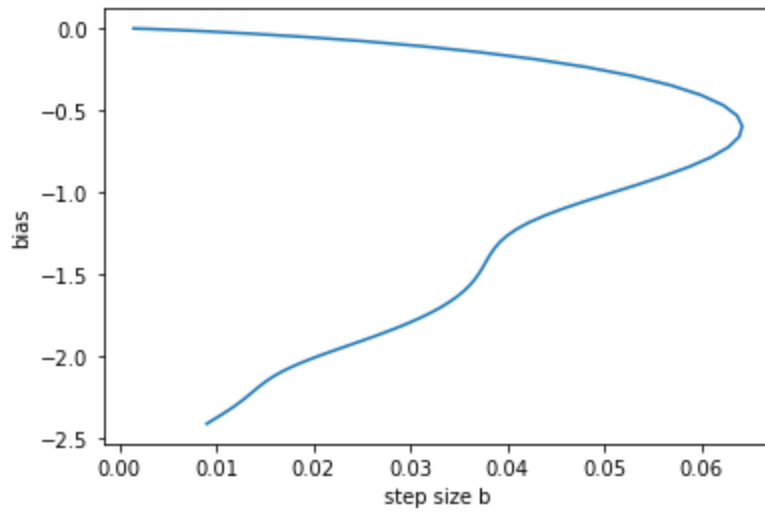


Fig 8. Step size for bias vs bias graph in Momentum-based Gradient Descent

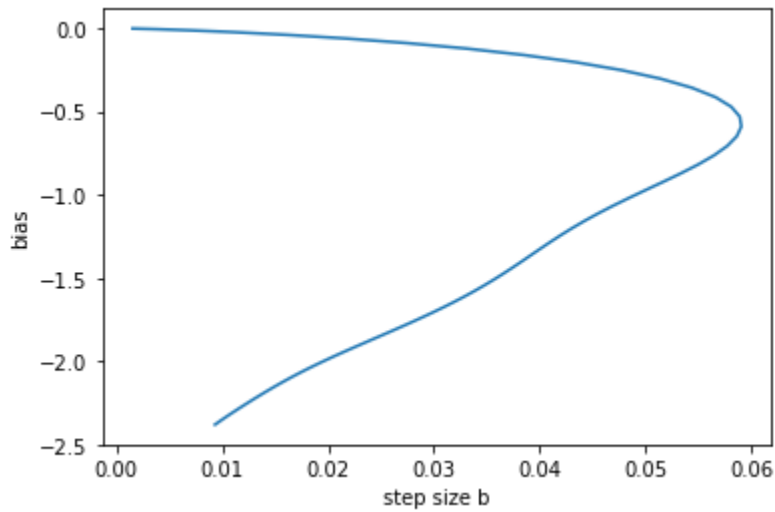


Fig 9. Step size for bias vs bias graph in Nesterov Accelerated Gradient Descent

Therefore, it can be inferred from fig 4-9, that the oscillations observed in the case of Nesterov Accelerated Gradient Descent are less than the ones observed in the case of Momentum-based Gradient Descent. Also, as Momentum-based Gradient Descent and Nesterov Accelerated Gradient Descent gives almost the output for the same function, training set, parameters, and hyper-parameters; it can be concluded that Nesterov Accelerated Gradient Descent is more efficient than Momentum-based Gradient Descent.