The term “error” used in this assignment/code, unless other specified, are all **L2-errors normalized by sample_size**
= np.linalg.norm(np.subtract(prediction,ground_truth))/sample_size

Question 2.a

Q: What do you find? How can this be remedied?
A: The error numpy.linalg.linalg.LinAlgError: Singular matrix popped out.
This is because that as the number of features increases, collinearity appears among the features, features are no longer independent. Therefore, the covariance matrix now has some of its rows filled with only 0’s (Singular matrix). The result of this singular matrix is that the X^TX is no longer inversible, thus a closed-form solution is not available, so the program cannot do the math anymore. We need to add regularization, in order to shift the singular values in the covariance matrix to some more positive values. This can be done through adding L1 regularization term or L2 regularization term.

Question 2.b

See **line#88** in script_regression.py

Question 2.c

Q: How does the result differ from (a)? Discuss the result in a couple of sentences
A: Now the program can successfully compute the result without crash. This is because the ridge regression adds a penalty term to regularize the . The program now is able to compute the inverse of X^TX without crashing. However, the result of ridge regression is biased.
For code, please see class ***RidgeLinearRegression*** in *regressionalgorithms.py* for detail. Please check the README.txt for any further notes on code.
Parameter: λ*=regwgt=0.01* `(average over 10 runs)`

```
Std error for RidgeLinearRegression 0.0008062659868634299
Average error for RidgeLinearRegression: 0.13206456802167582
```

Check the full execution log in *Appendix (c)*.

Question 2.d

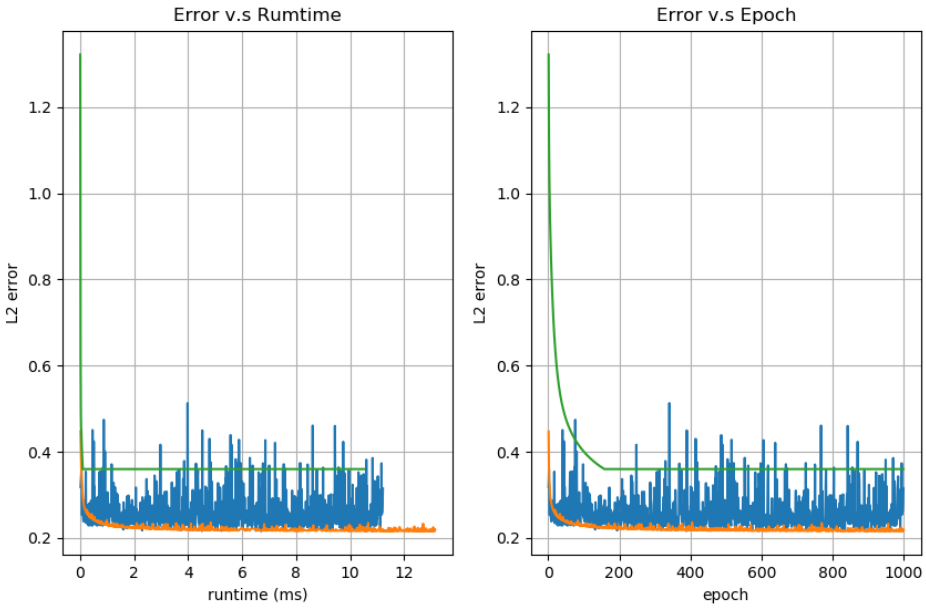
For code, please see class ***LassoRegression*** in *regressionalgorithms.py* for detail. Please check the README.txt for any further notes on code.
Parameter: λ*=regwgt=0.01* `(average over 10 runs)`

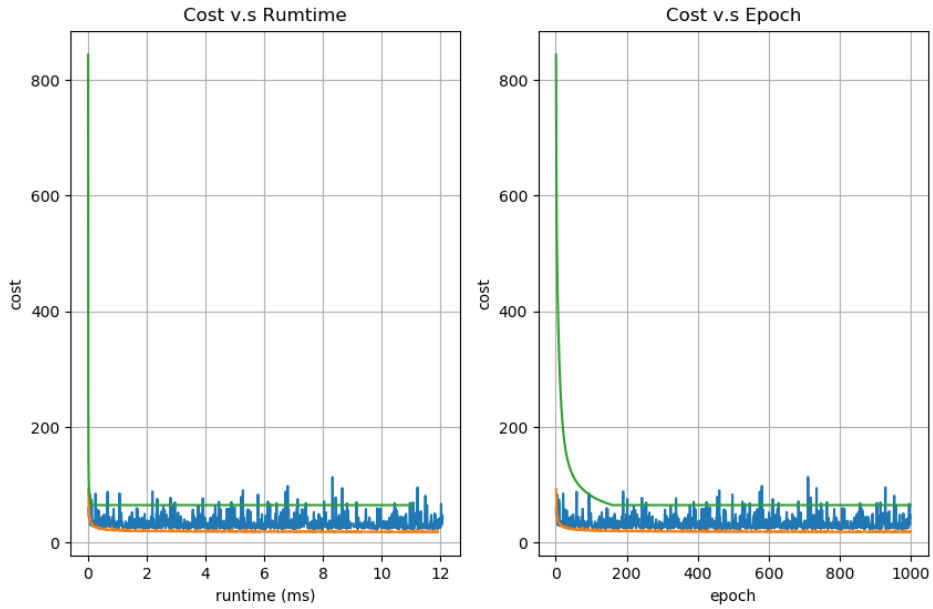
`Std error for LassoRegression 0.0008057103436139647`
`Average error for LassoRegression: 0.15952472434499443`
Check the full execution log in *Appendix (d)*.

Question 2.e

For code, please see class ***SGDLinearRegression*** in *regressionalgorithms.py* for detail. Please check the README.txt for any further notes on code.
Parameter: step_size=0.01, 1000 epochs `(average over 10 runs)`
`Std error for SGD 0.0062454916932106405`
`Average error for SGD: 0.17228029447246976`
Check the full execution log in *Appendix (e)*.

Question 2.f





Blue Curve: Stochastic Gradient Descent, step size=0.01
Parameter: step_size=0.01, 1000 epochs
Green curve: Batch Gradient Descent
Parameter: step_size=0.01, 1000 epoch, line search decay=0.7
Orange Curve (Extra after parameter tweaking): Stochastic Gradient Descent, step size=0.001
Parameter: step_size=0.001, 1000 epochs

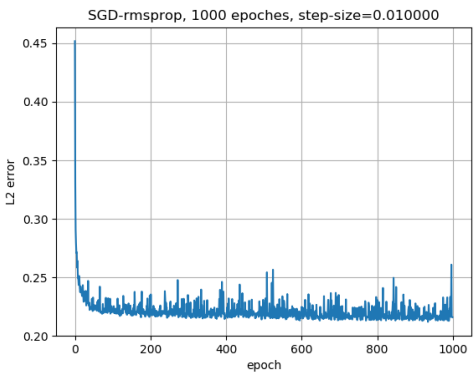
Q: Comparing SGD(1000 epochs, step-size=0.01) with Batch GD(1000 epochs, line search), in terms of the number of times the entire training set is processed
A: The SGD lowers down the cost within the very first epoch of training, the Batch GD lowers down the cost after 76 epochs (l2-error= 0.2950108255597983). It is because SGD uses online learning, it updates its weight *num_sample* times in a single training epoch, while the Batch GD starts with a random initial point and only updates once in an epoch. However, the SGD is always fluctuating and never converges to the local minimum, because of its constant learning rate of 0.01.
Note: The extra Orange Curve is the SGD with a lower step size 0.001. it is hard to tell from the plot that SGD(step size=0.01) is decreasing its cost/error. This is because it has a too large step-size.

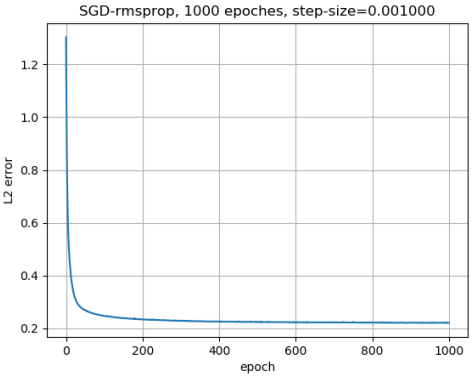
The **error versus epoch** and the **error versus runtime** for both algorithms are reported in the plot above.

For code, please see class ***BatchGDLinearRegression*** and ***SGDLinearRegression*** in *regressionalgorithms.py* for detail. Please check the README.txt for any further notes on code. If you need to re-generate the plot from my code, please uncomment the relevant code blocks.

Question Bonus a)

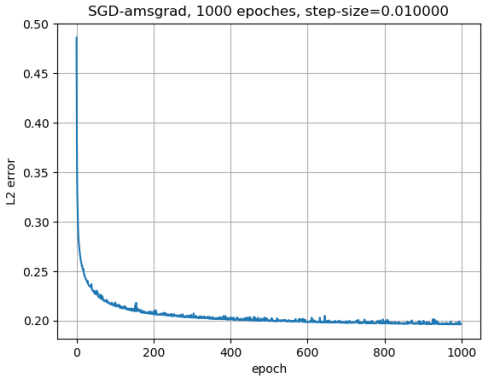
SGD with RMSprop
Parameter: decay rate=0.9, initial step size=0.01
The L2-error over the training set after 1000 epochs is: `0.16926827952381518`
The average standard error over the test test out of 10 runs under this parameter: `0.00613935386506591`
A much better convergence is resulted with a smaller initial step-size of 0.001. See below right.

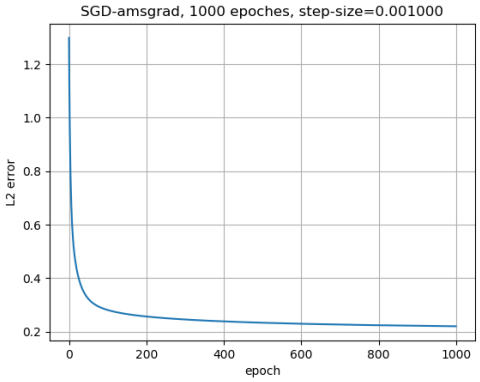




Question Bonus b)

SGD with Amsgrad
Parameter: decay rate=0.9, initial step size=0.01, beta1=0.9, beta2=0.99
The L2-error over the training set after 1000 epochs is: `0.1506555706092275`
The average standard error over the test test out of 10 runs under this parameter: `0.0012383612964978314`
A much better convergence is resulted with a smaller step-size of 0.001. See below right.





Appendix Overall （numruns=1）

```
Std error for Random 0.0
Average error for Random: 0.7051157243517832
Std error for Mean 0.0
Average error for Mean: 0.3169655483017599
Std error for FSLinearRegression5 0.0
Average error for FSLinearRegression5: 0.48437973292345915
Std error for FSLinearRegression50 0.0
Average error for FSLinearRegression50: 0.2051702122086488
Std error for RidgeLinearRegression 0.0
Average error for RidgeLinearRegression: 0.1320062019872152
Std error for LassoRegression 0.0
Average error for LassoRegression: 0.16242776331671915
Std error for SGD 0.0
Average error for SGD: 0.15068102377811923
Std error for BGD 0.0
Average error for BGD: 0.1650937240993985
Std error for SGDLinearRegressionRmsprop 0.0
Average error for SGDLinearRegressionRmsprop: 0.14382295891094118
Std error for SGDLinearRegressionAmsgrad 0.0
Average error for SGDLinearRegressionAmsgrad: 0.13271177285911373
```

Appendix (c)

```
Running on train=1000 and test=5000 samples for run 0
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.13143581472097784
Running on train=1000 and test=5000 samples for run 1
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.1307989274581724
Running on train=1000 and test=5000 samples for run 2
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.1325458127042246
Running on train=1000 and test=5000 samples for run 3
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.13362542629123553
Running on train=1000 and test=5000 samples for run 4
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.13006450838156217
Running on train=1000 and test=5000 samples for run 5
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.12740302846693777
Running on train=1000 and test=5000 samples for run 6
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
```

```
Error for RidgeLinearRegression: 0.12984123618313476
Running on train=1000 and test=5000 samples for run 7
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.1363080813651537
Running on train=1000 and test=5000 samples for run 8
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.13519570858323224
Running on train=1000 and test=5000 samples for run 9
Running learner = RidgeLinearRegression on parameters {'regwgt': 0.01}
Error for RidgeLinearRegression: 0.13342713606212725
Std error for RidgeLinearRegression 0.0008062659868634299
Average error for RidgeLinearRegression: 0.13206456802167582
```

Appendix (d)

```
Running on train=1000 and test=5000 samples for run 0
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.15553313019510168
Running on train=1000 and test=5000 samples for run 1
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.1579438210082077
Running on train=1000 and test=5000 samples for run 2
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.16153913794131605
Running on train=1000 and test=5000 samples for run 3
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.1573442147230973
Running on train=1000 and test=5000 samples for run 4
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.16054927363303884
Running on train=1000 and test=5000 samples for run 5
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.1572658828716155
Running on train=1000 and test=5000 samples for run 6
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.15855561794703718
Running on train=1000 and test=5000 samples for run 7
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.16414697195720568
Running on train=1000 and test=5000 samples for run 8
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.16254577149326657
Running on train=1000 and test=5000 samples for run 9
Running learner = LassoRegression on parameters {'regwgt': 0.01}
Error for LassoRegression: 0.1598234216800577
```

```
Std error for LassoRegression 0.0008057103436139647
Average error for LassoRegression: 0.15952472434499443
```

Appendix (e)

```
Running on train=1000 and test=5000 samples for run 0
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.1641009400771434
Running on train=1000 and test=5000 samples for run 1
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.1730663372955966
Running on train=1000 and test=5000 samples for run 2
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.16423486047263433
Running on train=1000 and test=5000 samples for run 3
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.16400542433003754
Running on train=1000 and test=5000 samples for run 4
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.16851627674704311
Running on train=1000 and test=5000 samples for run 5
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.23042801429212145
Running on train=1000 and test=5000 samples for run 6
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.1620198498435328
Running on train=1000 and test=5000 samples for run 7
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.1640072951651359
Running on train=1000 and test=5000 samples for run 8
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.17155991804976617
Running on train=1000 and test=5000 samples for run 9
Running learner = SGD on parameters {'num_epoch': 1000, 'stepsize': 0.01}
Error for SGD: 0.16086402845168615
Std error for SGD 0.0062454916932106405
Average error for SGD: 0.17228029447246976
```

Appendix Bonus

```
Std error for SGDLinearRegressionRmsprop 0.00613935386506591
Average error for SGDLinearRegressionRmsprop: 0.16926827952381518
Std error for SGDLinearRegressionAmsgrad 0.0012383612964978314
Average error for SGDLinearRegressionAmsgrad: 0.1506555706092275
```