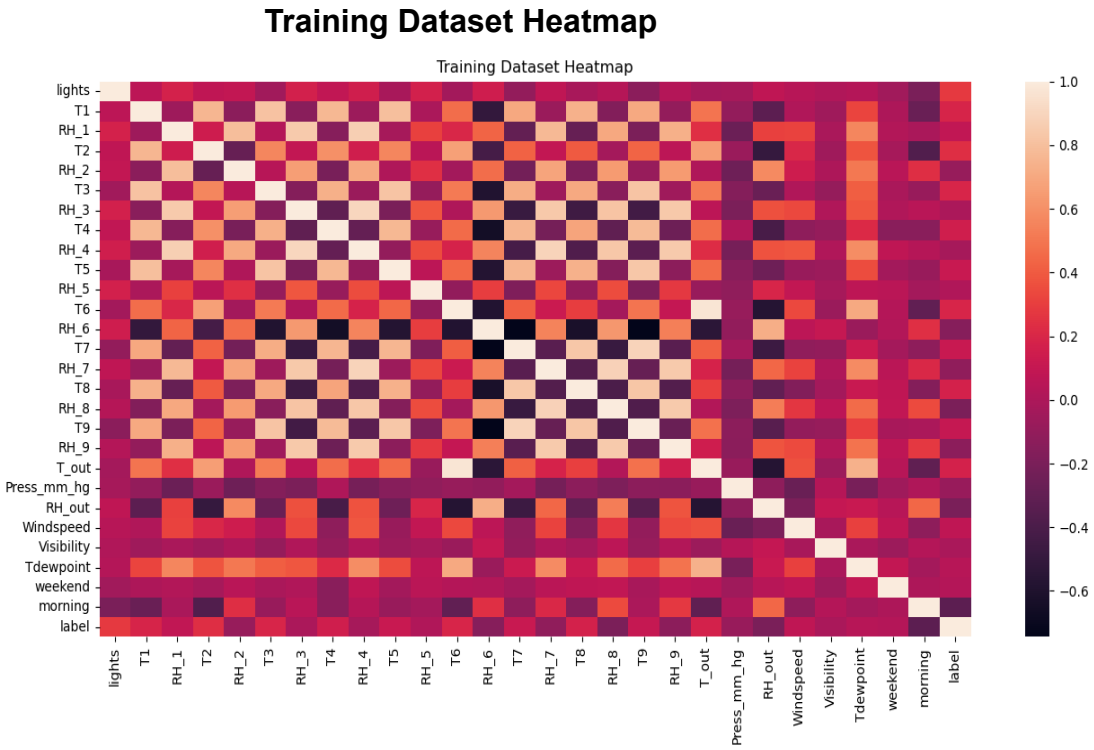
/* THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS OR LARGE LANGUAGE MODELS SUCH AS CHATGPT.
Tommy Skodje */

I collaborated with the following classmates for this homework:
None

1. a. A useful feature that could be extracted from the date and timestamp column could be whether that day was a weekend or not. On weekends people tend to be home more, so energy usage may look different than on a weekday.

   Another useful feature could be whether the time is in the morning or the afternoon. During the early hours of the morning, people may be asleep, which could change energy usage.

1. d.


Training Dataset Heatmap
Training Dataset Heatmap
lights
T1
RH_1
T2
RH_2
T3
RH_3
T4
RH_4
T5
RH_5
T6
RH_6
T7
RH_7
T8
RH_8
T9
RH_9
T_out
Press_mm_hg
RH_out
Windspeed
Visibility
Tdewpoint
weekend
morning
label
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

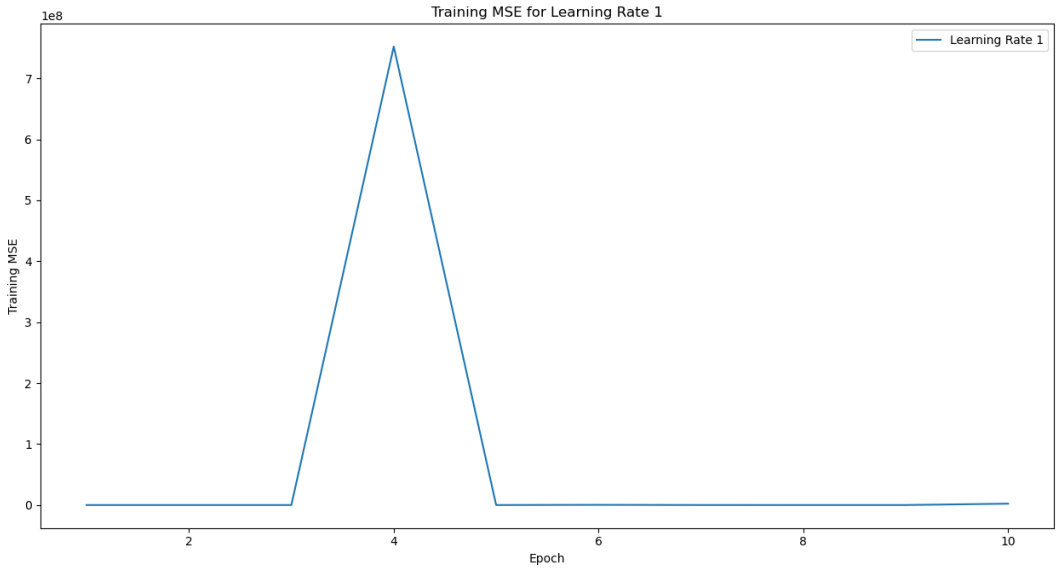
1. e. From the correlation plot, the features that have the highest correlation to the label are lights, T1, T2, T3, T4, T5, T6, T7, T8, T9, and T_out. However, T1 is also heavily correlated with T2, T3, T4, T5, T7, T8, and T9. We only need one of these variables, since they are so highly correlated with each other. I will select only T1 out of the set of T1, T2, T3, T4, T5, T7, T8, and T9. T1 is moderately correlated with T6 and T_out, but less so than the other features mentioned above, so I will use both T6 and T_out to train the linear regression model as well. Finally, lights is not heavily correlated with any of the other features, so I will be using it to train the linear regression model as well.

   In conclusion, the features I will be using to train the linear regression model are: lights, T1, T6, and T_out

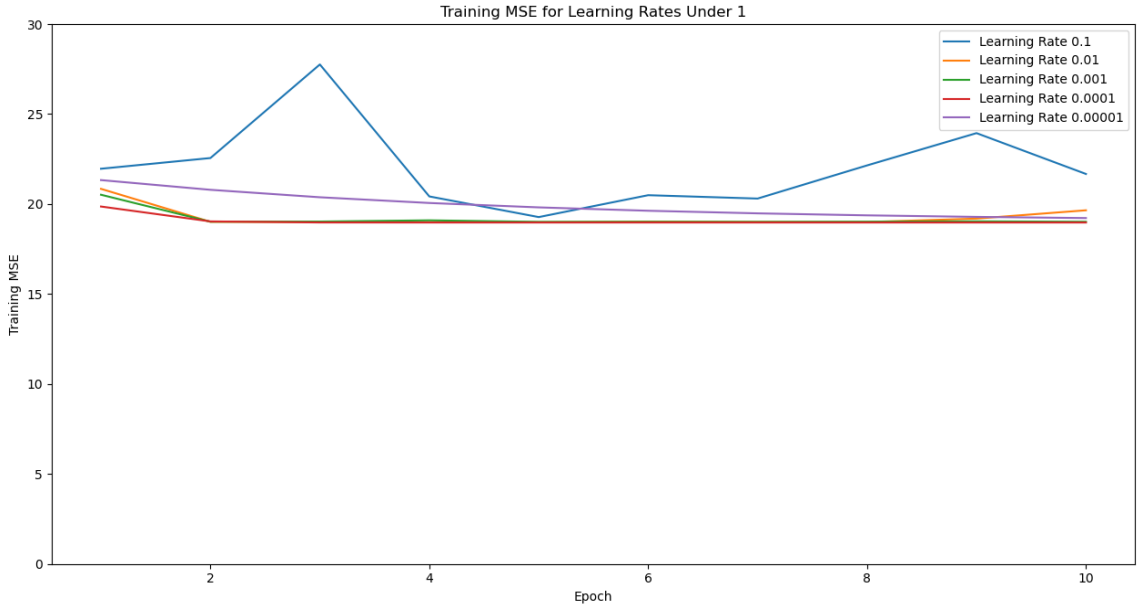
1. g. Another preprocessing step that needs to be done is to standardize the data. This is needed because the variables may be on different scales, which could cause some unexpected results when running a linear regression model. To fix this, we standardize the data. It's also important to standardize the test data with the same scaling attributes used for the training data. I will be using the sklearn StandardScaler in order to standardize the data.

3. c.

Below is the graph for learning rate = 1. The training MSE was so high for one of the epochs that I had to put this learning rate in another graph.



Below is the graph of the rest of the learning rates

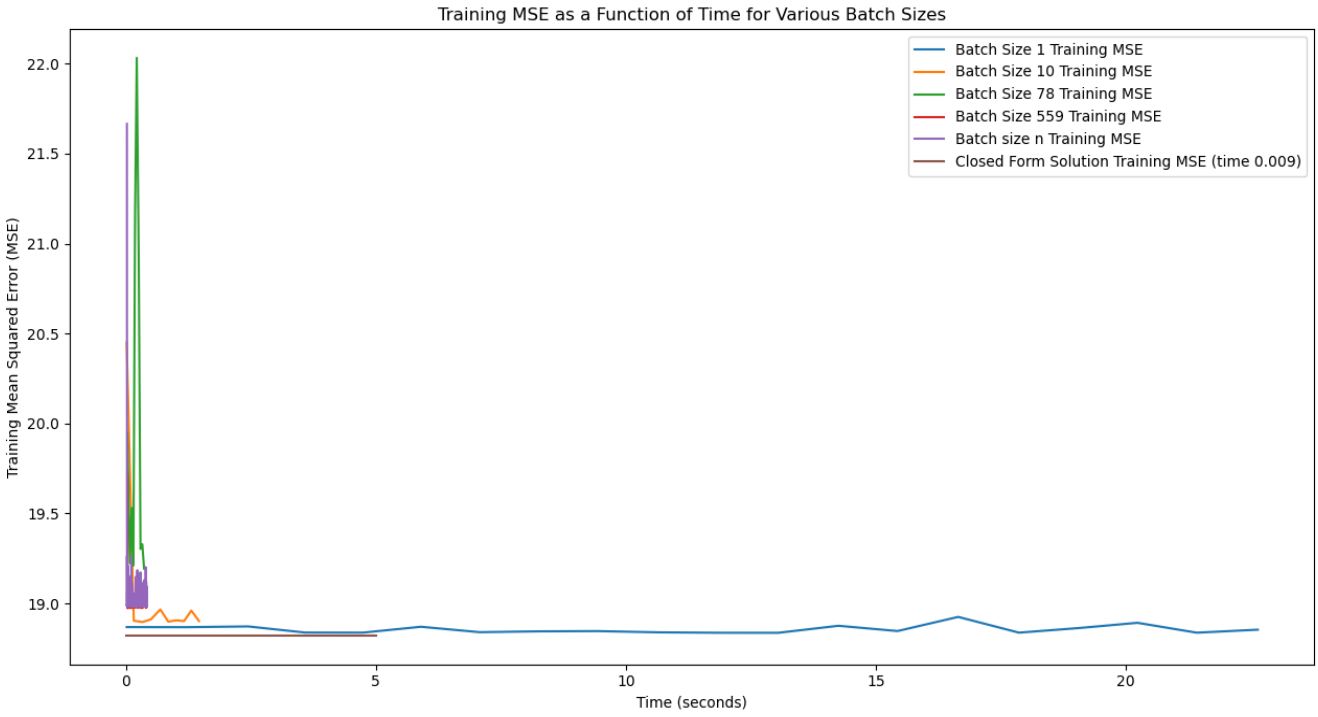


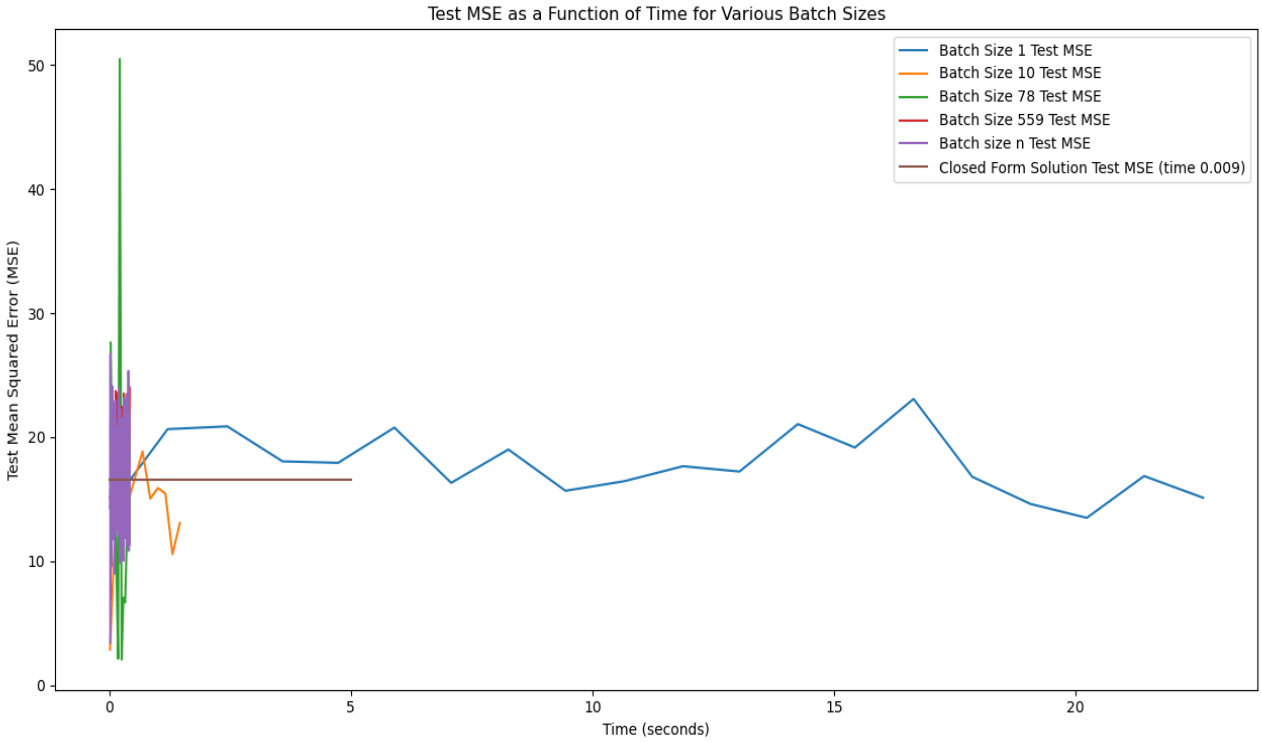
From the graphs, a training MSE of just under 20 seems to be where the learning rates converge to. It's clear that learning rate 1 is too high, as one of the epochs had a very

high training MSE, which none of the other learning rates had. It also never reached a training MSE under 20. A learning rate of 0.1 did not reach a training MSE under 20 after 10 epochs, so it seems like that learning rate is too high as well. The learning rate of 0.00001 took quite a bit longer than the other learning rates to reach a training MSE under 20, so it is likely too low of a learning rate. Out of the remaining learning rates of 0.01, 0.001, 0.0001, they all reach a training MSE under 20 relatively quickly, but the learning rate of 0.01 has a slight increase in training MSE at the 10th epoch. Learning rates 0.001 and 0.0001 are almost identical. Therefore, 0.001 is the ideal learning rate, since it converges to a low training MSE faster because it learns at a quicker rate.

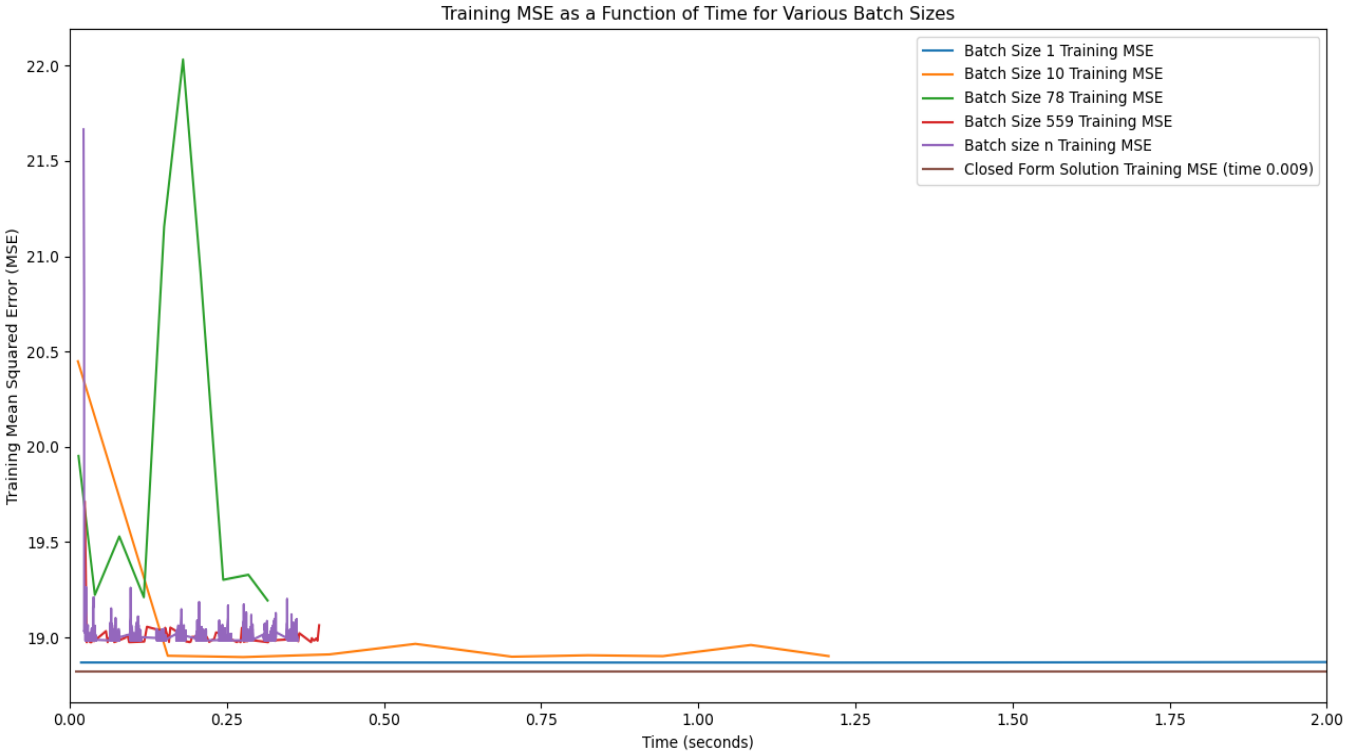
3. d. Below is the plot for the mean squared error on the training and test data as a function of the epoch for the ideal learning rate of 0.001

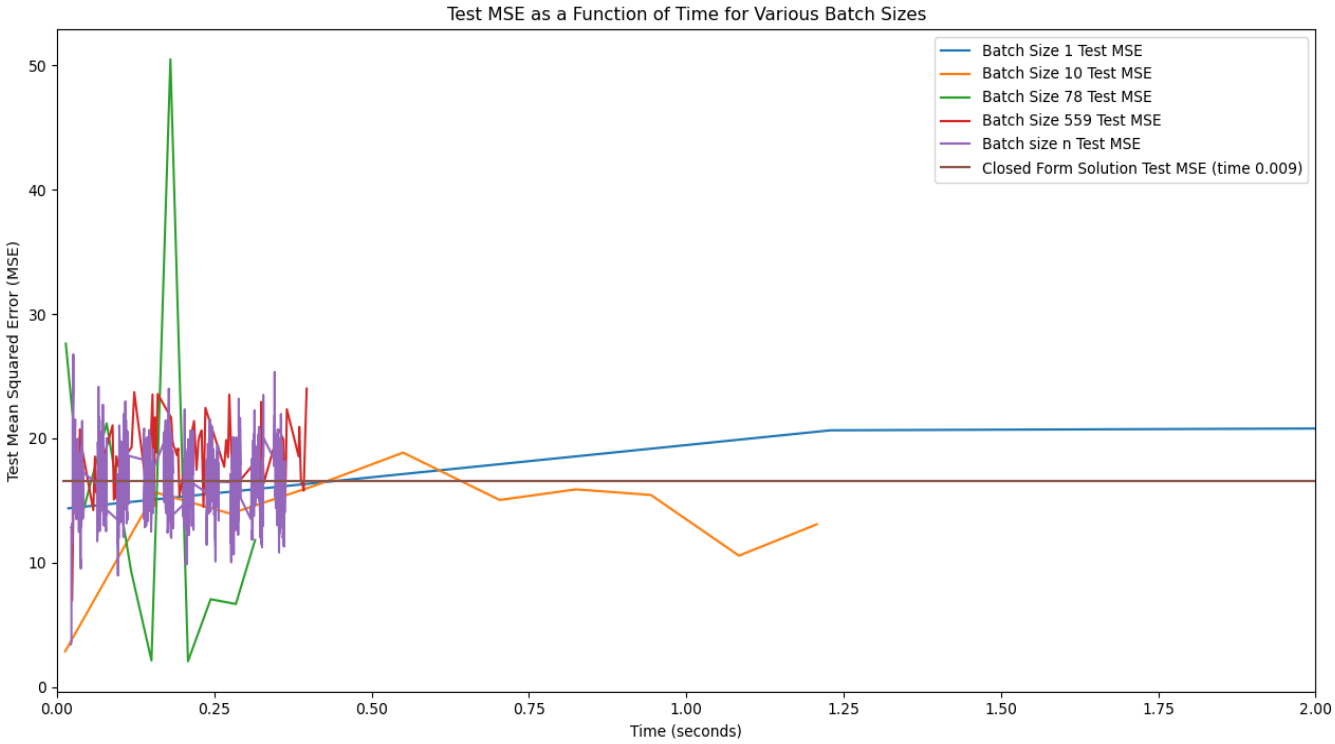
4. a. Below are graphs of the training and test MSE as a function of the total time for different batch sizes, where the training size is divisible by all of the batch sizes used. The brown line in each of the graphs represents the training and test MSE of the closed-form solution. The time taken for the closed-form solution was 0.009, but the line has been extended slightly for visibility.





Here are zoomed-in views of both of the graphs above.


Training MSE as a Function of Time for Various Batch Sizes
Batch Size 1 Training MSE
Batch Size 10 Training MSE
Batch Size 78 Training MSE
Batch Size 559 Training MSE
Batch size n Training MSE
Closed Form Solution Training MSE (time 0.009)
Training Mean Squared Error (MSE)
Time (seconds)
22.0
21.5
21.0
20.5
20.0
19.5
19.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Test MSE as a Function of Time for Various Batch Sizes
Batch Size 1 Test MSE
Batch Size 10 Test MSE
Batch Size 78 Test MSE
Batch Size 559 Test MSE
Batch size n Test MSE
Closed Form Solution Test MSE (time 0.009)
Test Mean Squared Error (MSE)
Time (seconds)
0
10
20
30
40
50
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

4. b. My observations based on the plots in 4a are as follows. First, the closed-form solution took significantly less time than any of the batch sizes for gradient descent, as it only took 0.009 seconds to find a closed-form solution.

Next, it seems like as the batch size increases, the time taken for the gradient descent algorithm decreases, with batch size = n having the lowest time taken out of all of the gradient descent algorithms. This trend holds for both the training and test data.

As for performance on the training data, the closed-form solution performed the best. For the training data, larger batch sizes performed worse in regards to MSE, with batch size = n having the highest training MSE.

However, for performance on the test data, the closed-form solution performed only about average, performing about on-par with batch size = n. Batch size = 10 performed the best on the test data, followed by batch size = 78. Batch size = 559 performed the worst.

It seems like the trade-offs between the different batch sizes are between performance and time. Lower batch sizes seem to perform better by producing lower training and test MSEs, but they also take longer to complete. Higher batch sizes don't perform as well, but they run much faster. There seems to be a "sweet spot" for batch size for this set of data at around batch size = 10. It produces performance about equal to or even better than batch size = 1 while taking significantly less time to compute than batch size = 1.

There is a similar trade-off between gradient descent and the closed-form solution. The closed-form solution is the faster algorithm, but it does not produce the absolute lowest test MSE. It performs better than higher batch sizes, but does not perform as well lower batch sizes. The closed-form solution also has an additional downside of not working for datasets that do not have a closed-form solution.