CMPE 462 Assignment 1 Report

Part 1

I have implemented a single function taking batch_size parameter to solve both step 1 and step2.

- When batch_size=np.inf or batch_size>=sample_count it works as full gradient descent
- When batch_size=1 it works as stochastic gradient descent
- When 1<batch_size<sample_count it works as mini batch gradient descent

That function also have an optional step_size parameter. And when I tried a few values, I found out that if I keep it somewhat bigger than 1e-5 (such as 1e-4) its loss value is increased on each iteration rather than decreasing and quickly arrives to infinity. Thus, I have chosen small, medium, big step_size values as 1e-7, 1e-6, and 1e-5 respectively for testing and kept the medium one as default.

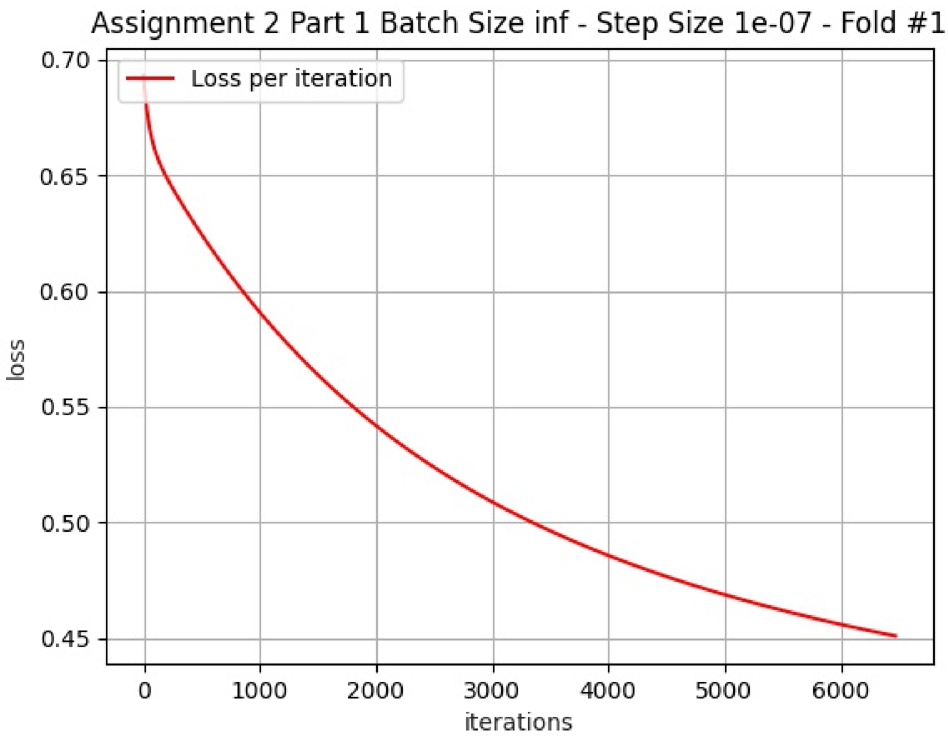
Applied 5-fold cross-validation, collected loss over iteration plots for each of them. Calculated error in cross-validation via the "simple error formula" shown in Lec04-LogisticRegression.pdf page 22-23 rather than the loss function used elsewhere. Collected train and test errors for each fold, and their average.

Below are cross-validation results and loss over iteration plots for each batch_size x step_size x fold combination.

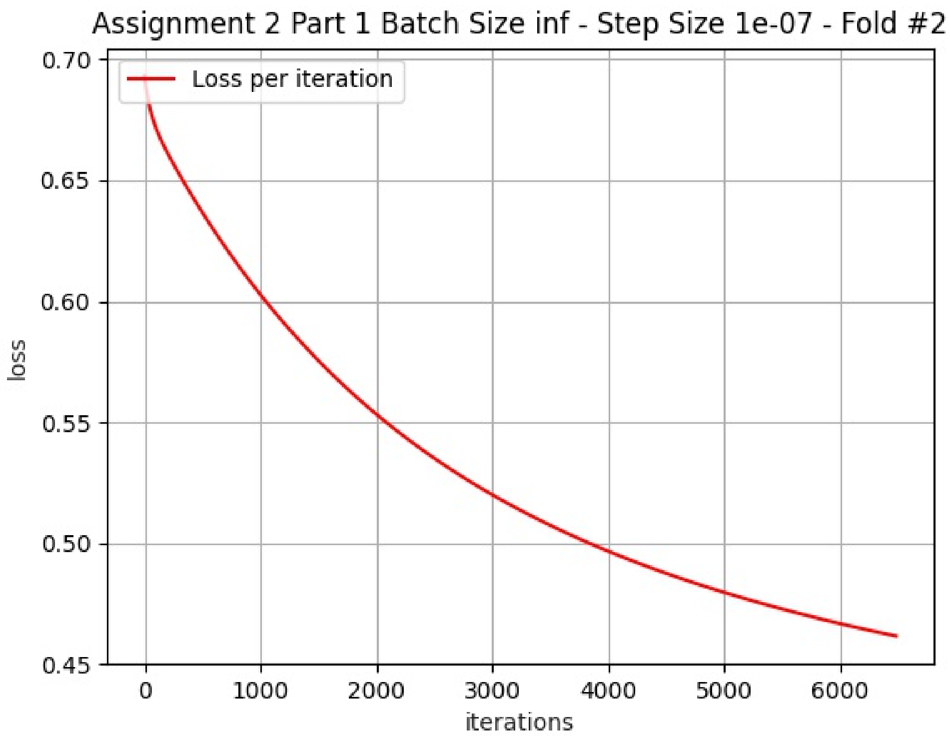
Batch Size inf (FGD) - Step Size 1e-7 (Small)

Average err_train = 0.15055732515442818
Average err_test = 0.15261621809682047
Stats by Fold:
Fold #1 : iterations = 6473.0 , err_train = 0.145 , err_test = 0.178
Fold #2 : iterations = 6485.0 , err_train = 0.149 , err_test = 0.155
Fold #3 : iterations = 6358.0 , err_train = 0.150 , err_test = 0.153
Fold #4 : iterations = 6576.0 , err_train = 0.149 , err_test = 0.150
Fold #5 : iterations = 6050.0 , err_train = 0.160 , err_test = 0.128

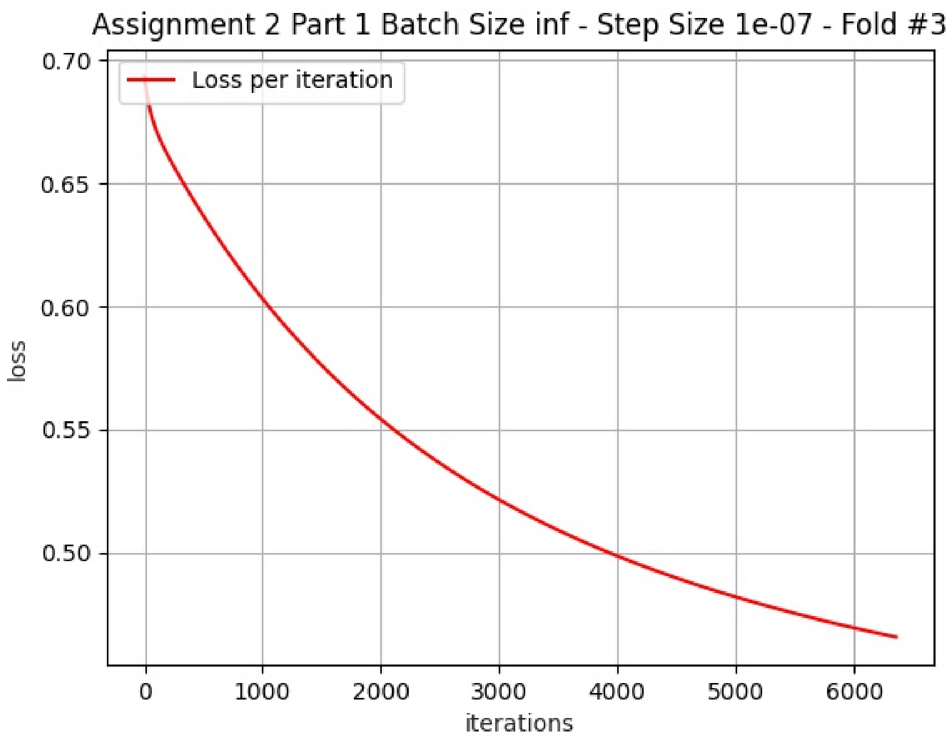
Batch Size inf (FGD) - Step Size 1e-7 (Small) - Fold #1



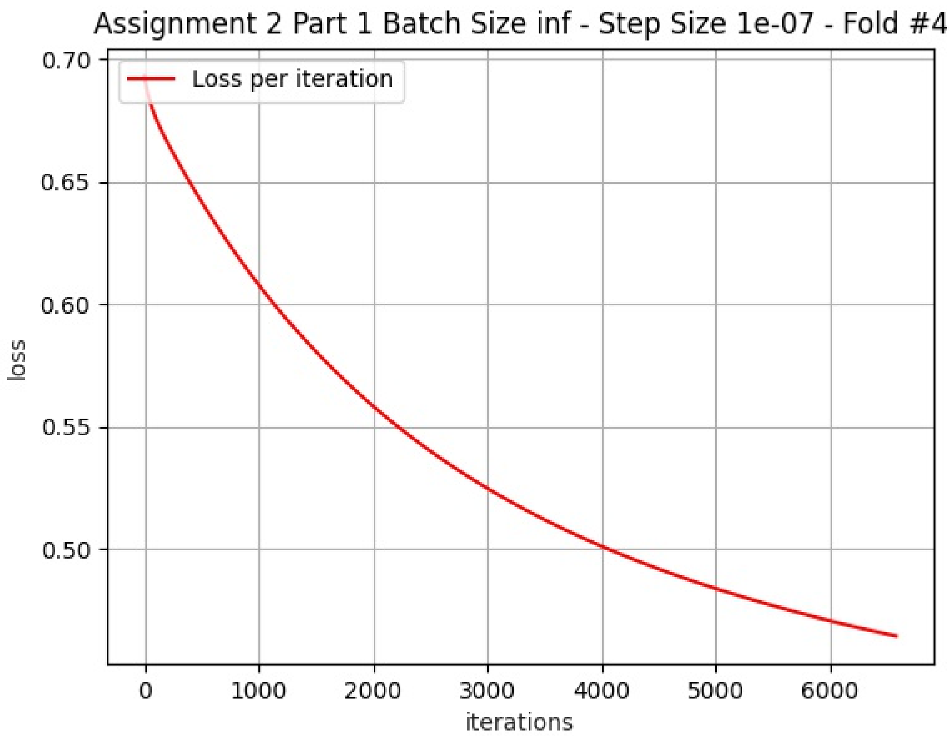
Batch Size inf (FGD) - Step Size 1e-7 (Small) - Fold #2



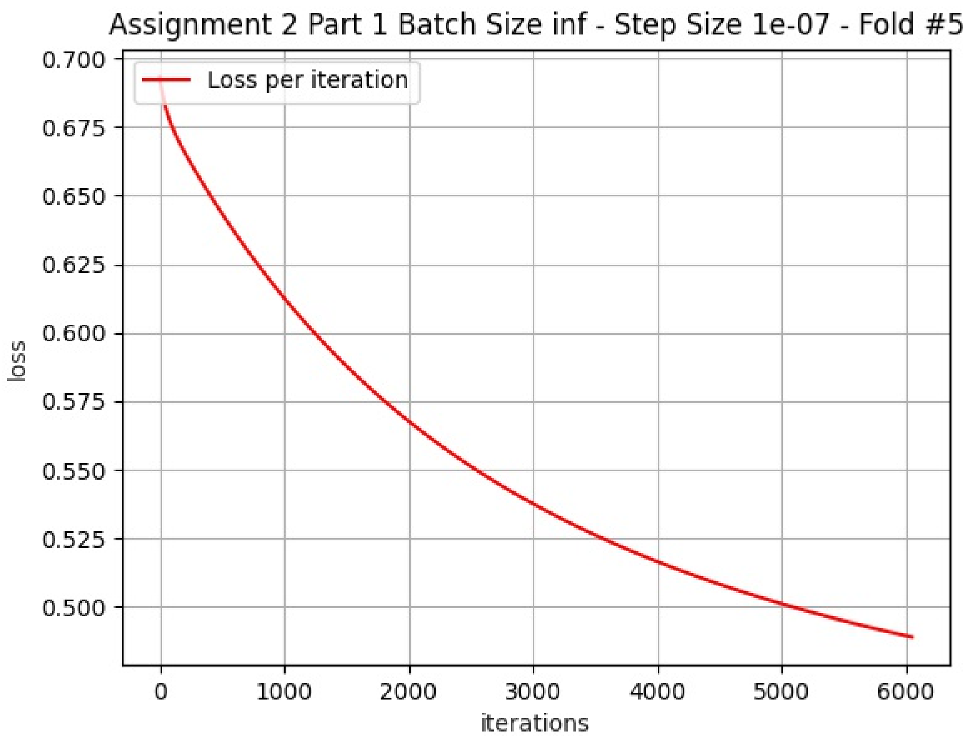
Batch Size inf (FGD) - Step Size 1e-7 (Small) - Fold #3



Batch Size inf (FGD) - Step Size 1e-7 (Small) - Fold #4



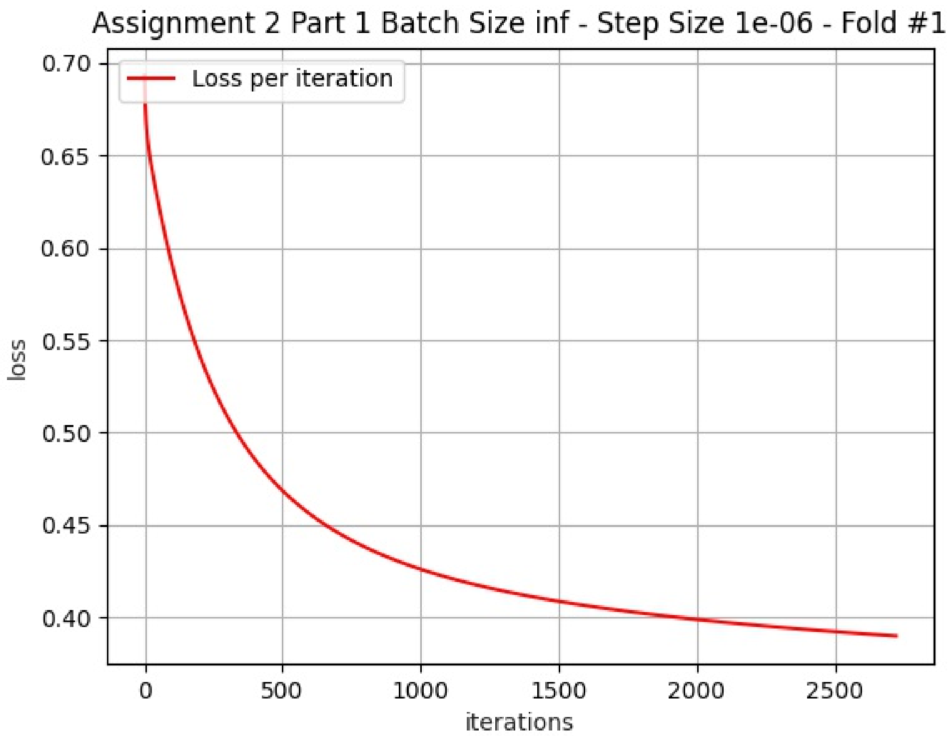
Batch Size inf (FGD) - Step Size 1e-7 (Small) - Fold #5



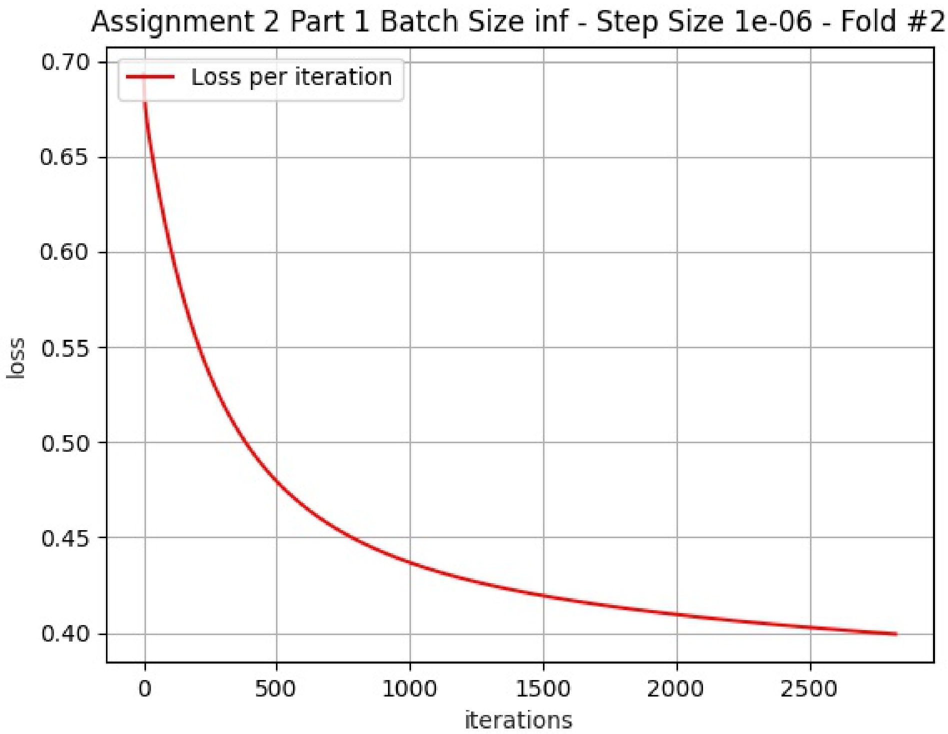
Batch Size inf (FGD) - Step Size 1e-6 (Medium)

Average err_train = 0.13103323850981358
Average err_test = 0.13333879646475127
Stats by Fold:
Fold #1 : iterations = 2724.0 , err_train = 0.126 , err_test = 0.159
Fold #2 : iterations = 2825.0 , err_train = 0.130 , err_test = 0.136
Fold #3 : iterations = 2678.0 , err_train = 0.131 , err_test = 0.135
Fold #4 : iterations = 3038.0 , err_train = 0.127 , err_test = 0.138
Fold #5 : iterations = 2685.0 , err_train = 0.141 , err_test = 0.098

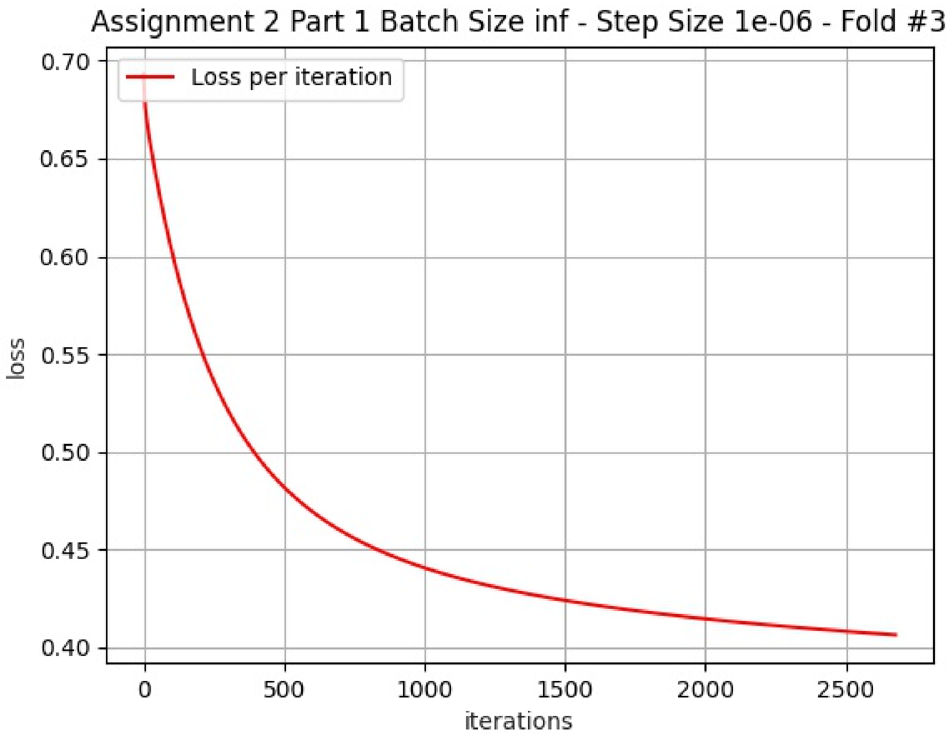
Batch Size inf (FGD) - Step Size 1e-6 (Medium) - Fold #1



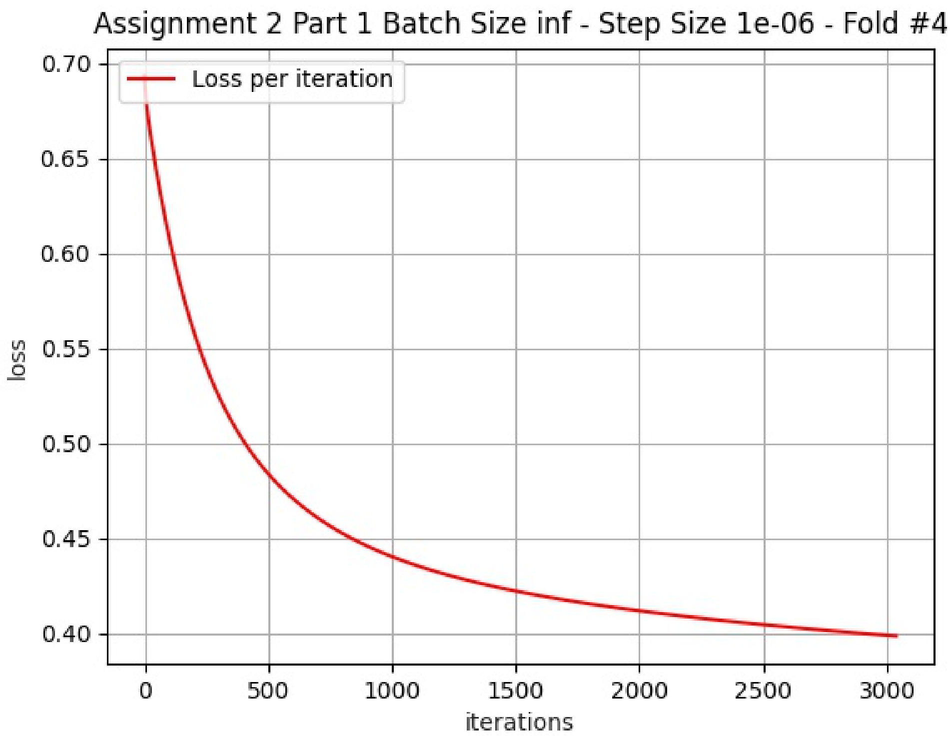
Batch Size inf (FGD) - Step Size 1e-6 (Medium) - Fold #2



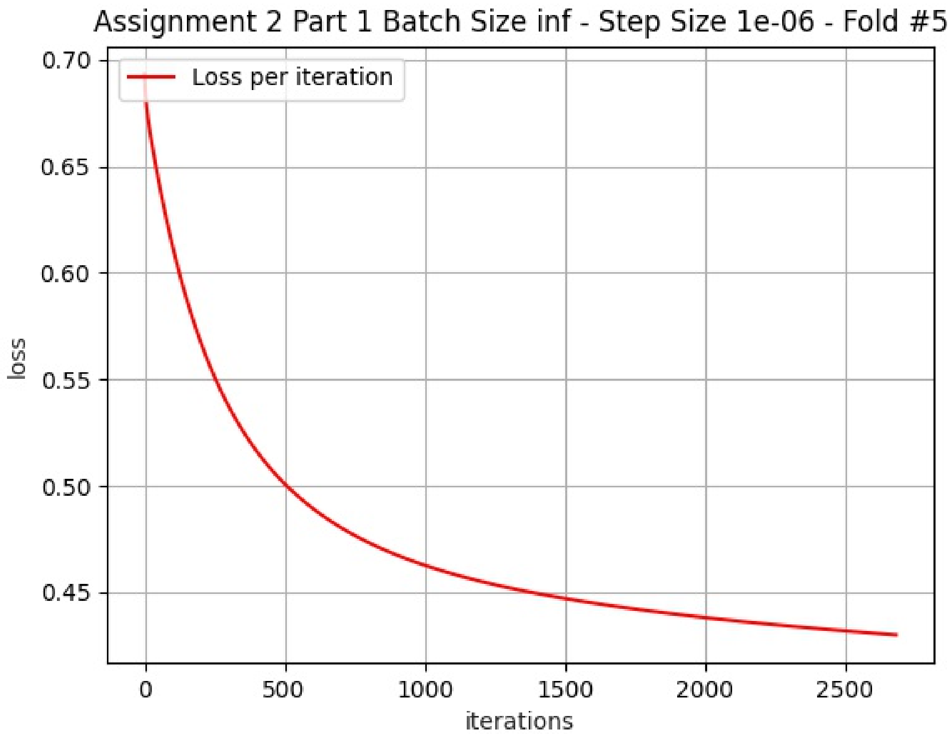
Batch Size inf (FGD) - Step Size 1e-6 (Medium) - Fold #3



Batch Size inf (FGD) - Step Size 1e-6 (Medium) - Fold #4



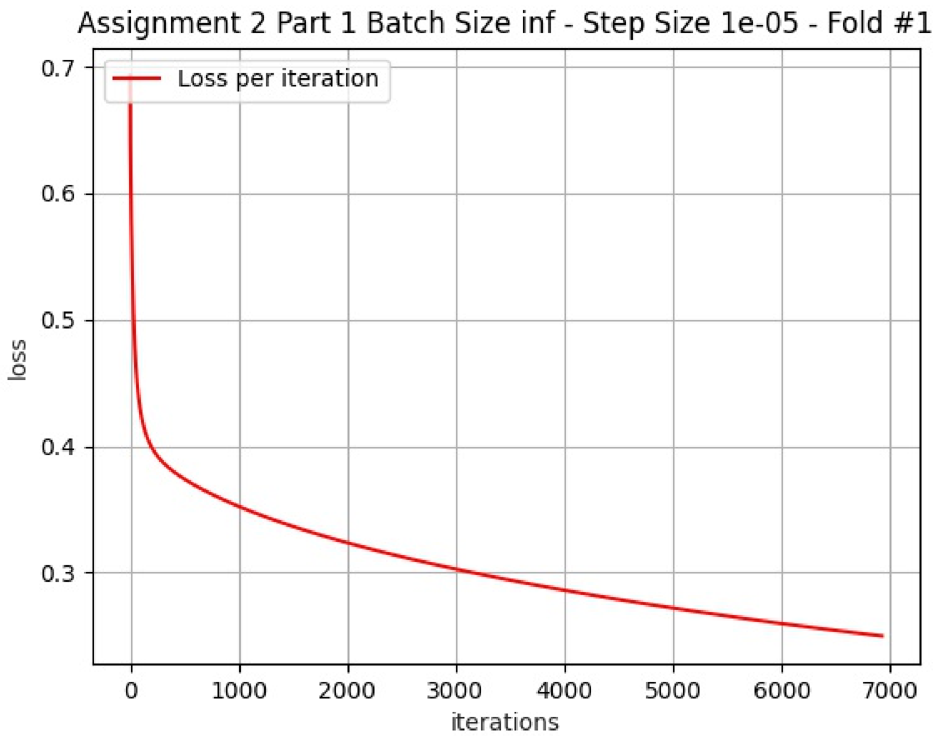
Batch Size inf (FGD) - Step Size 1e-6 (Medium) - Fold #5



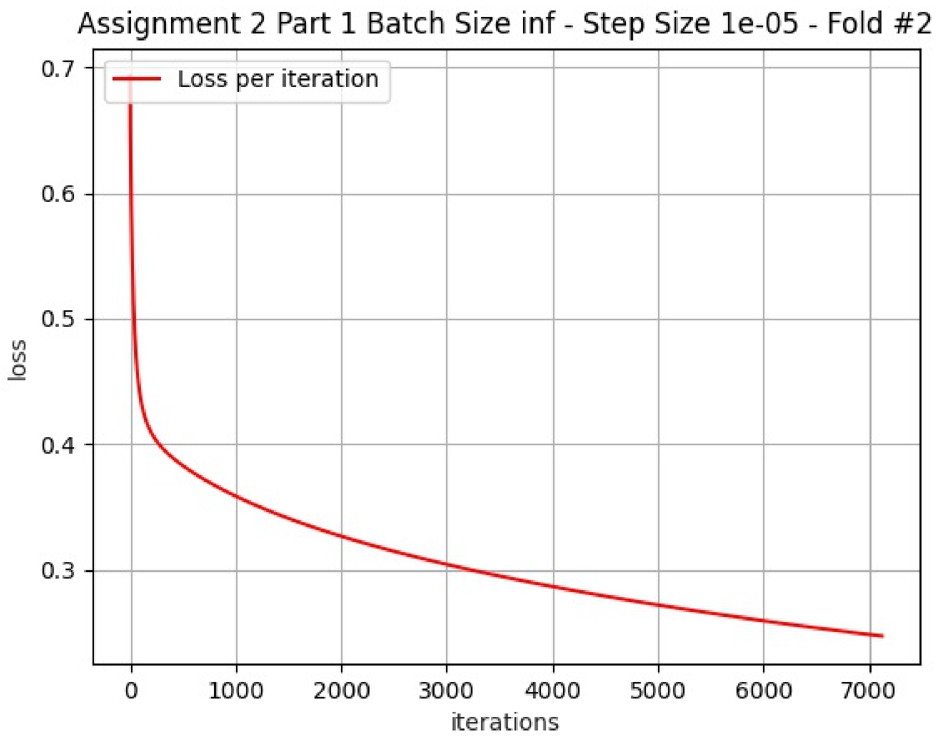
Batch Size inf (FGD) - Step Size 1e-5 (Big)

Average err_train = 0.07207749844544428
Average err_test = 0.07694272484894403
Stats by Fold:
Fold #1 : iterations = 6927.0 , err_train = 0.074 , err_test = 0.087
Fold #2 : iterations = 7126.0 , err_train = 0.072 , err_test = 0.090
Fold #3 : iterations = 7552.0 , err_train = 0.072 , err_test = 0.067
Fold #4 : iterations = 7162.0 , err_train = 0.070 , err_test = 0.096
Fold #5 : iterations = 7998.0 , err_train = 0.073 , err_test = 0.045

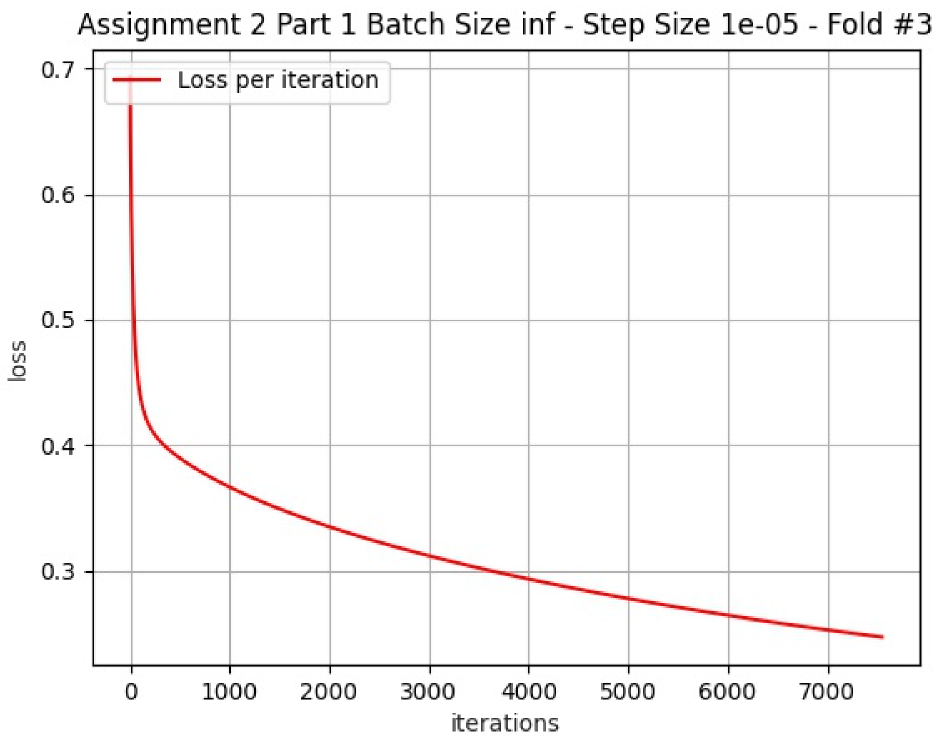
Batch Size inf (FGD) - Step Size 1e-5 (Big) - Fold #1



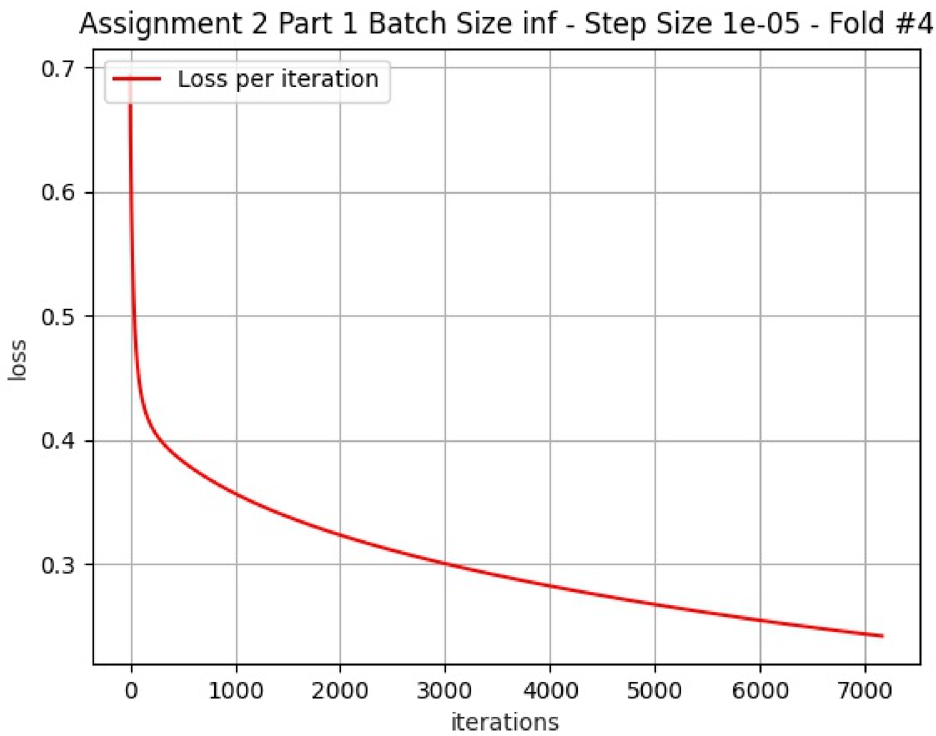
Batch Size inf (FGD) - Step Size 1e-5 (Big) - Fold #2



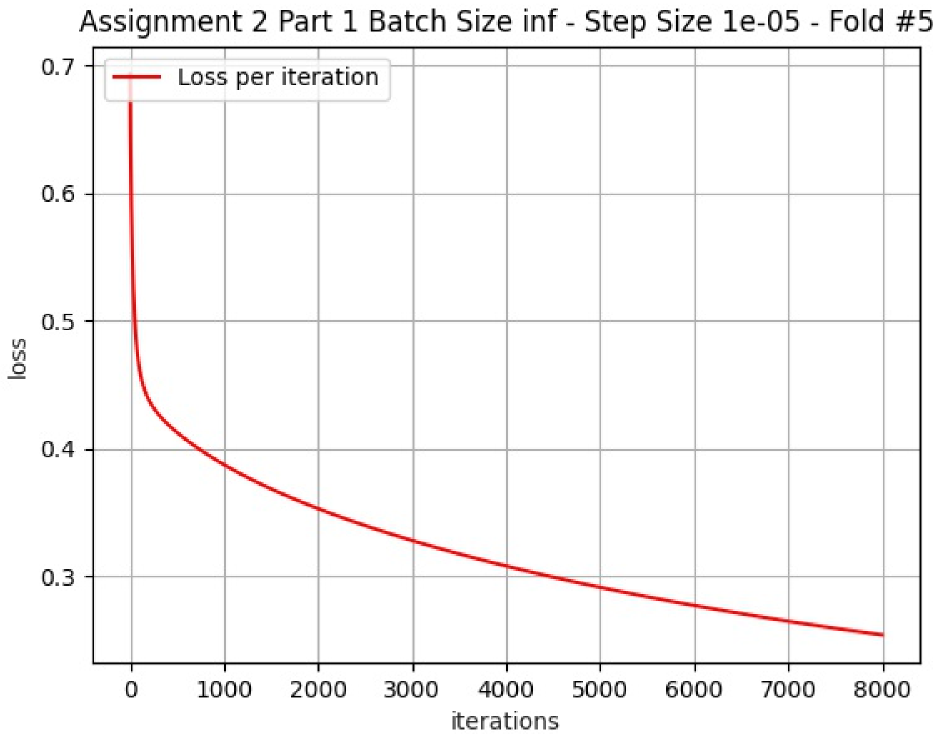
Batch Size inf (FGD) - Step Size 1e-5 (Big) - Fold #3



Batch Size inf (FGD) - Step Size 1e-5 (Big) - Fold #4



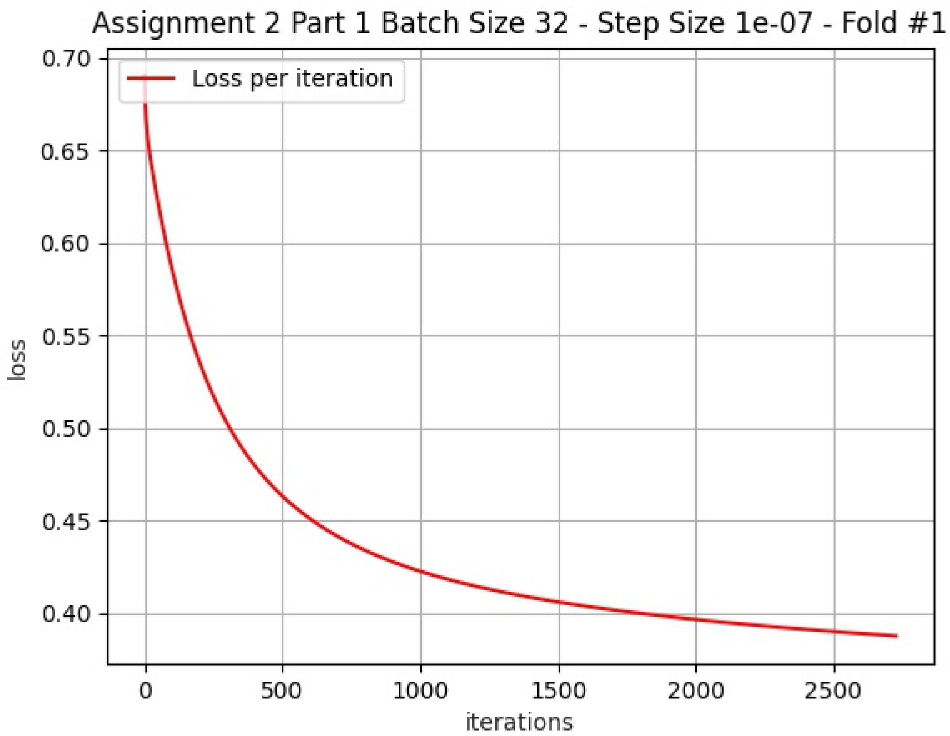
Batch Size inf (FGD) - Step Size 1e-5 (Big) - Fold #5



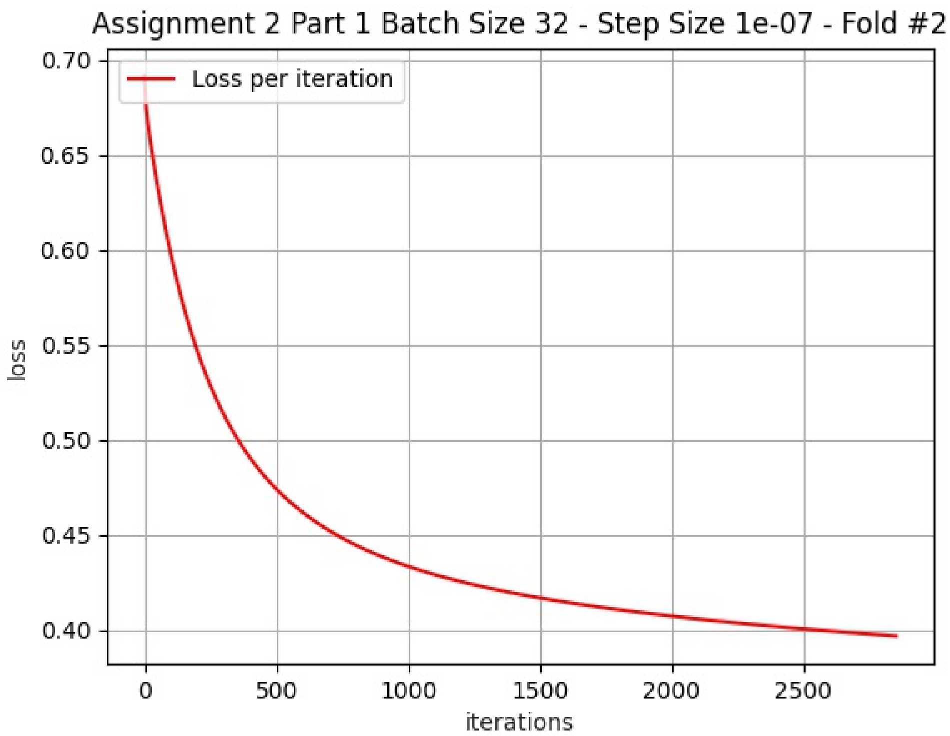
Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small)

Average err_train = 0.1301209234288046
Average err_test = 0.13237757609349568
Stats by Fold:
Fold #1 : iterations = 2728.0 , err_train = 0.126 , err_test = 0.159
Fold #2 : iterations = 2853.0 , err_train = 0.129 , err_test = 0.135
Fold #3 : iterations = 2694.0 , err_train = 0.130 , err_test = 0.134
Fold #4 : iterations = 3095.0 , err_train = 0.126 , err_test = 0.137
Fold #5 : iterations = 2751.0 , err_train = 0.139 , err_test = 0.097

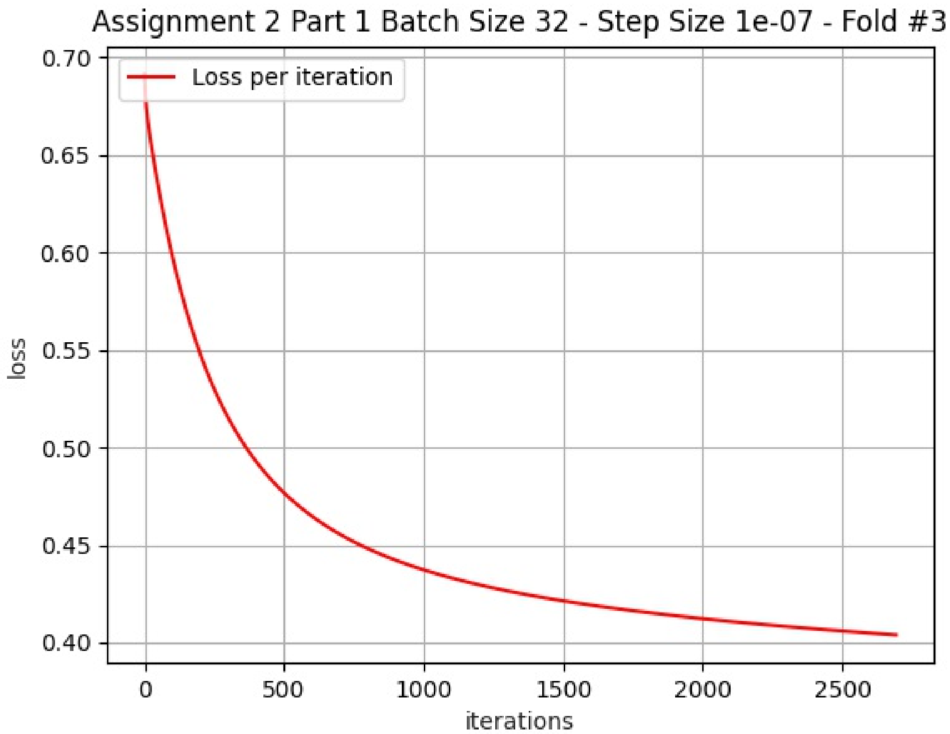
Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small) - Fold #1



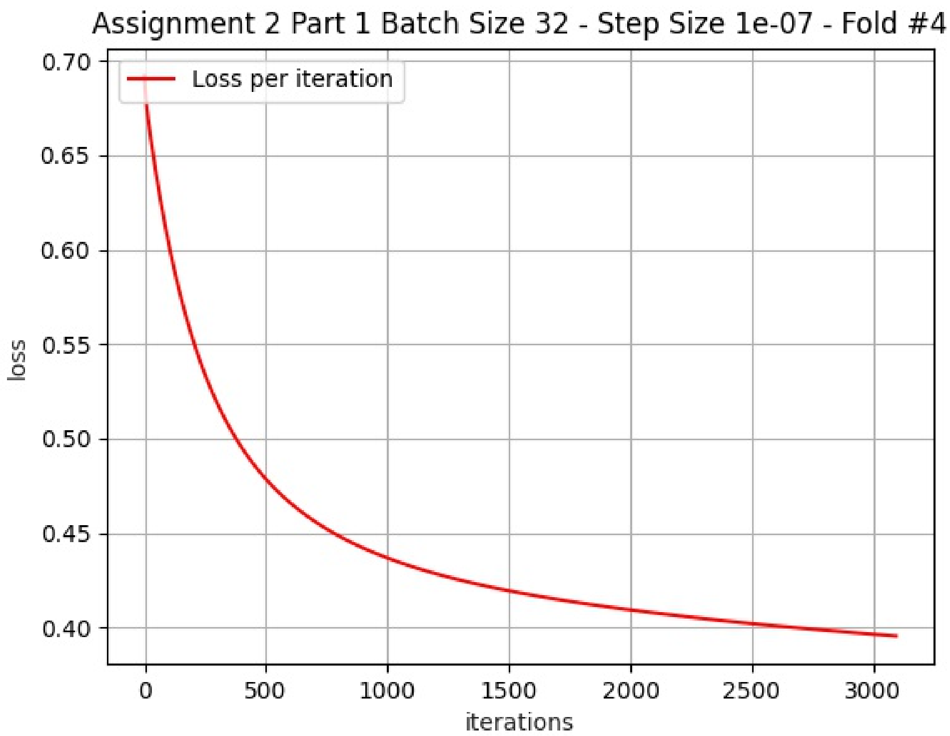
Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small) - Fold #2



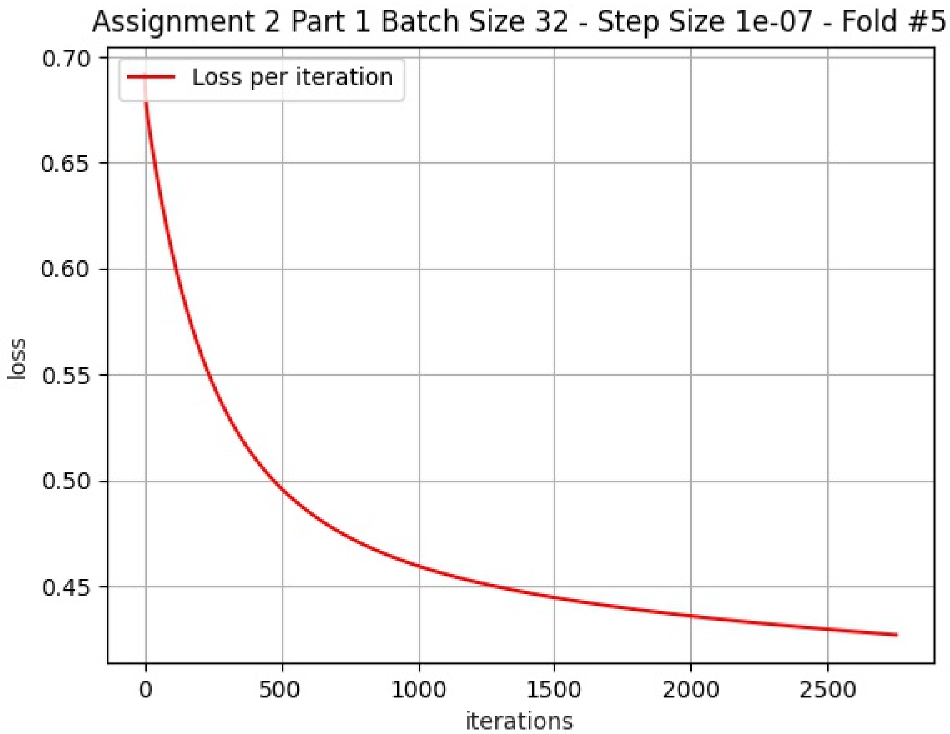
Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small) - Fold #3



Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small) - Fold #4



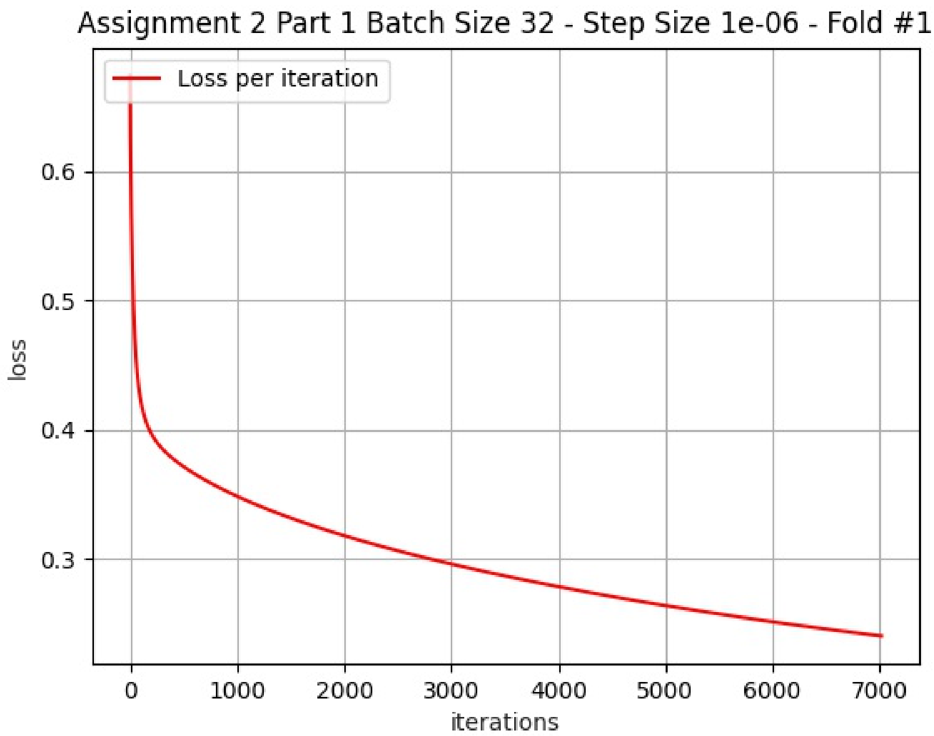
Batch Size 32 (MiniBatch) - Step Size 1e-7 (Small) - Fold #5



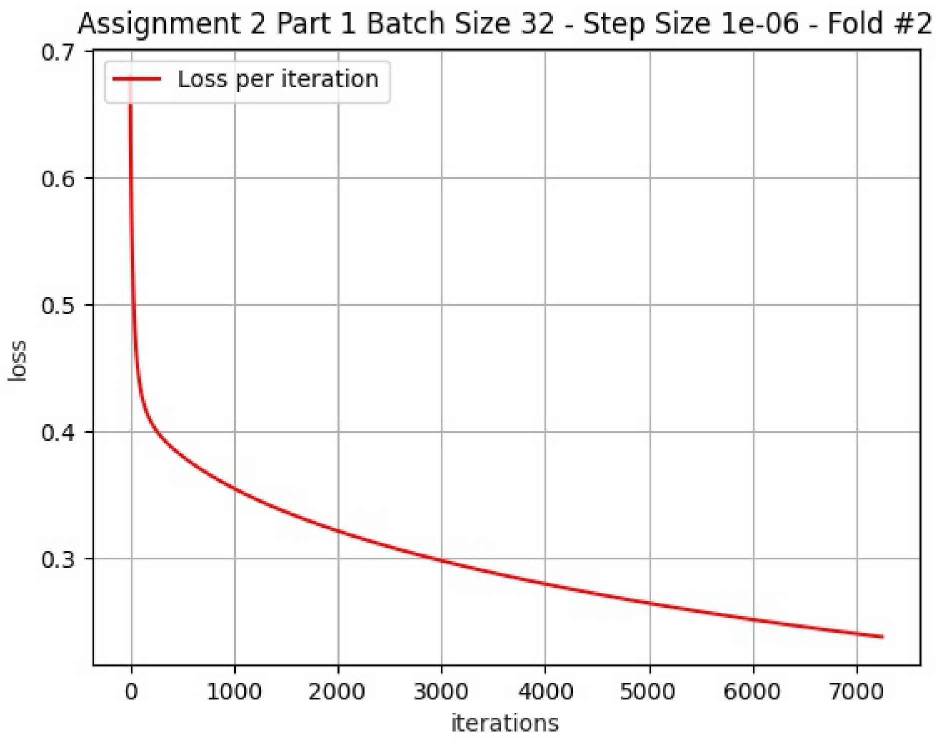
Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium)

Average err_train = 0.06850718108092783
Average err_test = 0.0733356749825757
Stats by Fold:
Fold #1 : iterations = 7020.0 , err_train = 0.070 , err_test = 0.084
Fold #2 : iterations = 7248.0 , err_train = 0.068 , err_test = 0.085
Fold #3 : iterations = 7596.0 , err_train = 0.068 , err_test = 0.064
Fold #4 : iterations = 7228.0 , err_train = 0.066 , err_test = 0.092
Fold #5 : iterations = 7958.0 , err_train = 0.070 , err_test = 0.043

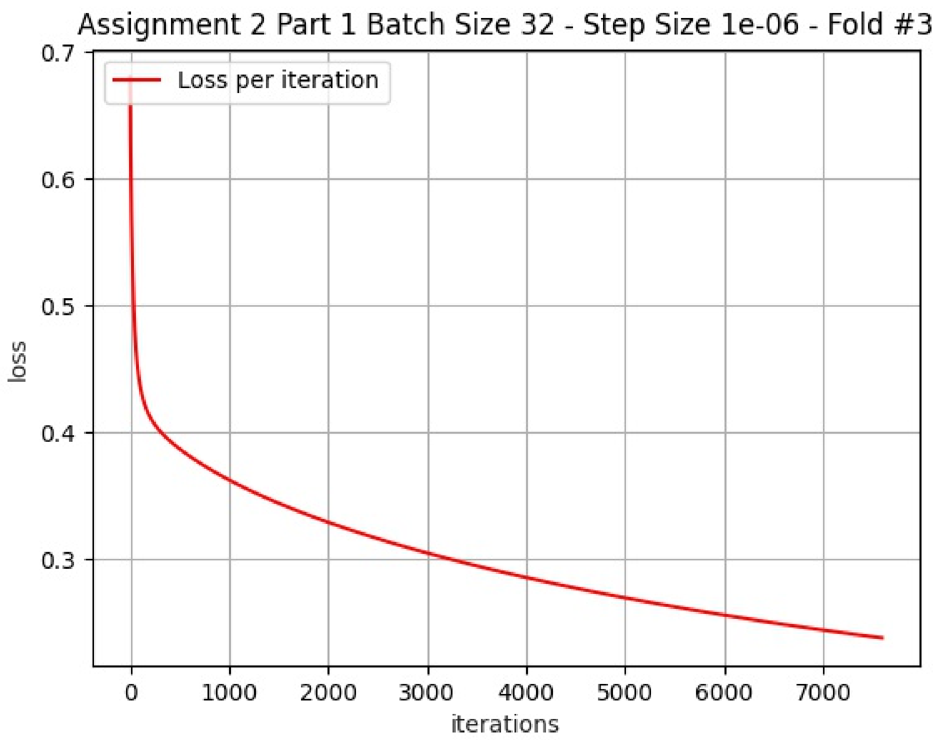
Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium) - Fold #1



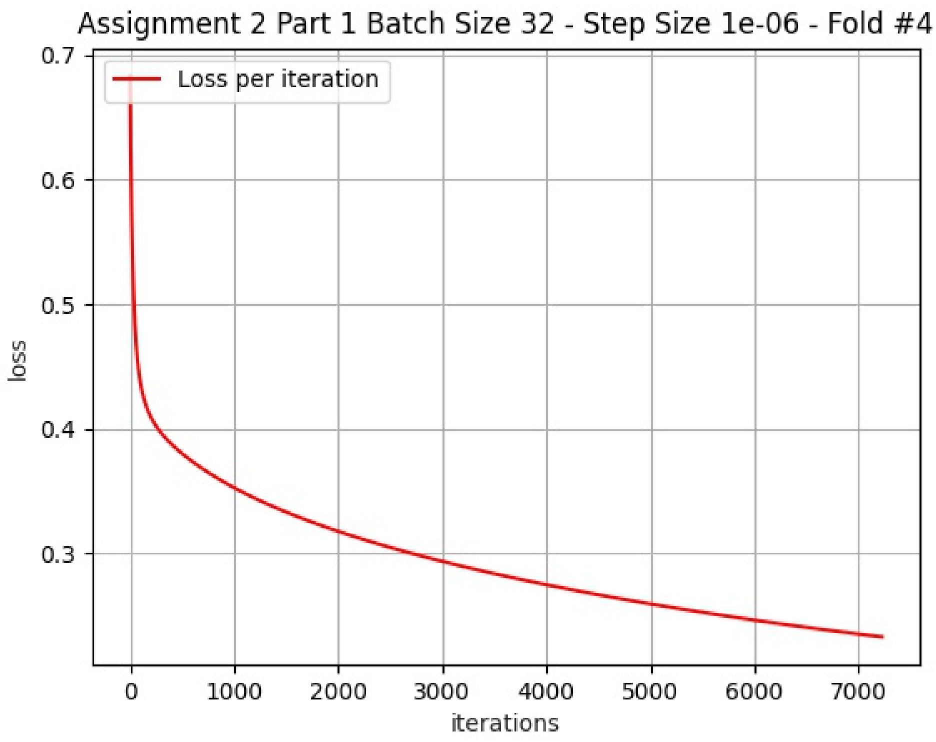
Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium) - Fold #2



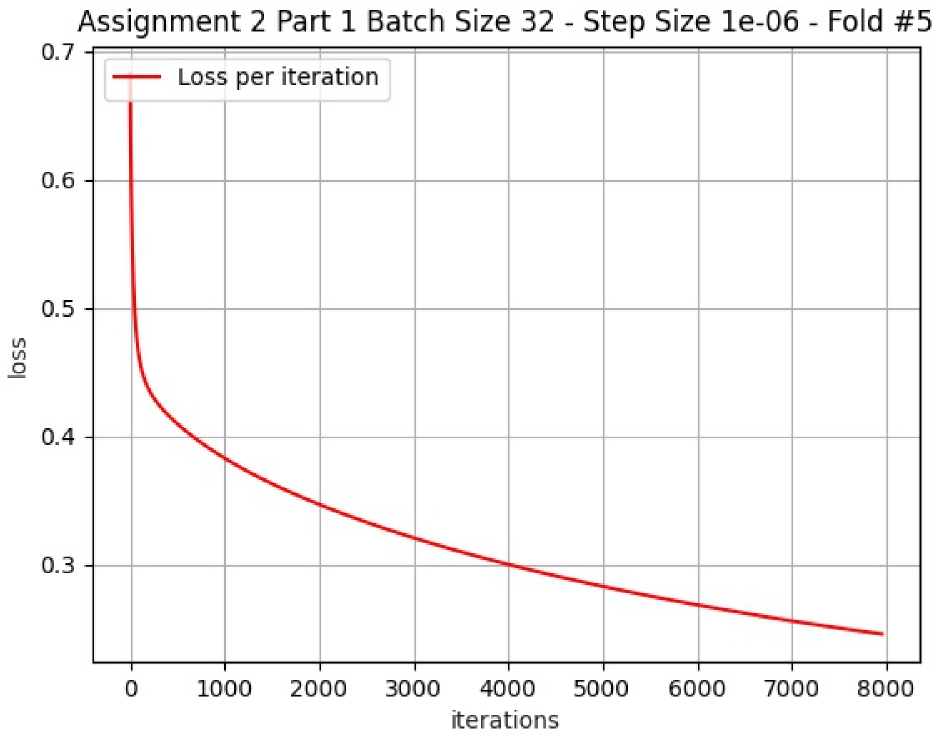
Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium) - Fold #3



Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium) - Fold #4



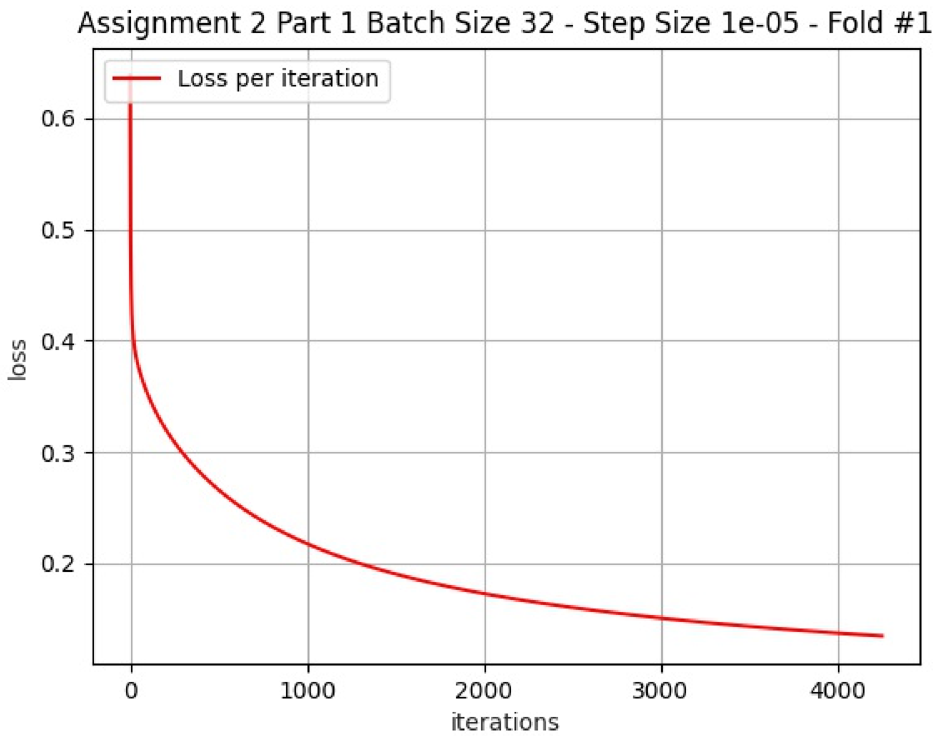
Batch Size 32 (MiniBatch) - Step Size 1e-6 (Medium) - Fold #5



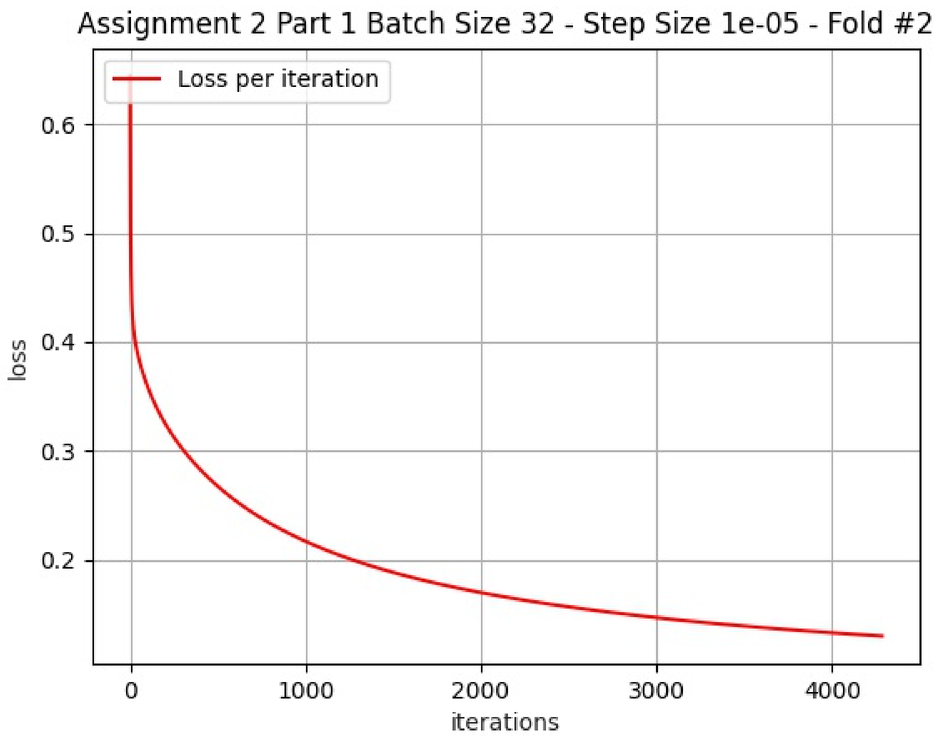
Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big)

Average err_train = 0.03502782140028424
Average err_test = 0.04064824741488453
Stats by Fold:
Fold #1 : iterations = 4251.0 , err_train = 0.036 , err_test = 0.040
Fold #2 : iterations = 4287.0 , err_train = 0.034 , err_test = 0.045
Fold #3 : iterations = 4362.0 , err_train = 0.035 , err_test = 0.037
Fold #4 : iterations = 4268.0 , err_train = 0.033 , err_test = 0.054
Fold #5 : iterations = 4352.0 , err_train = 0.037 , err_test = 0.027

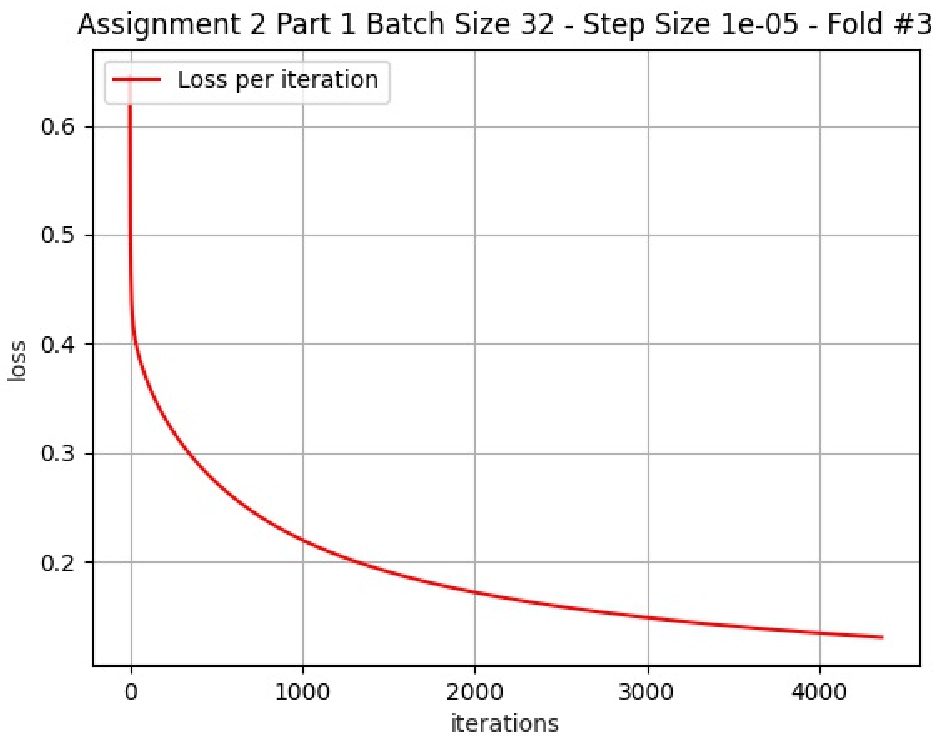
Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big) - Fold #1



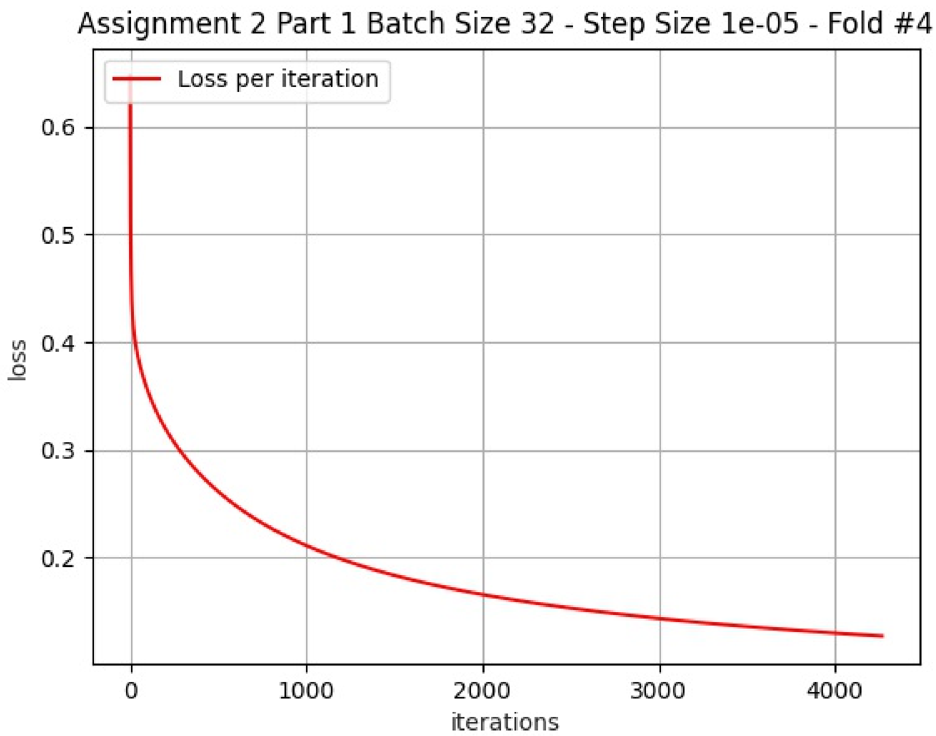
Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big) - Fold #2



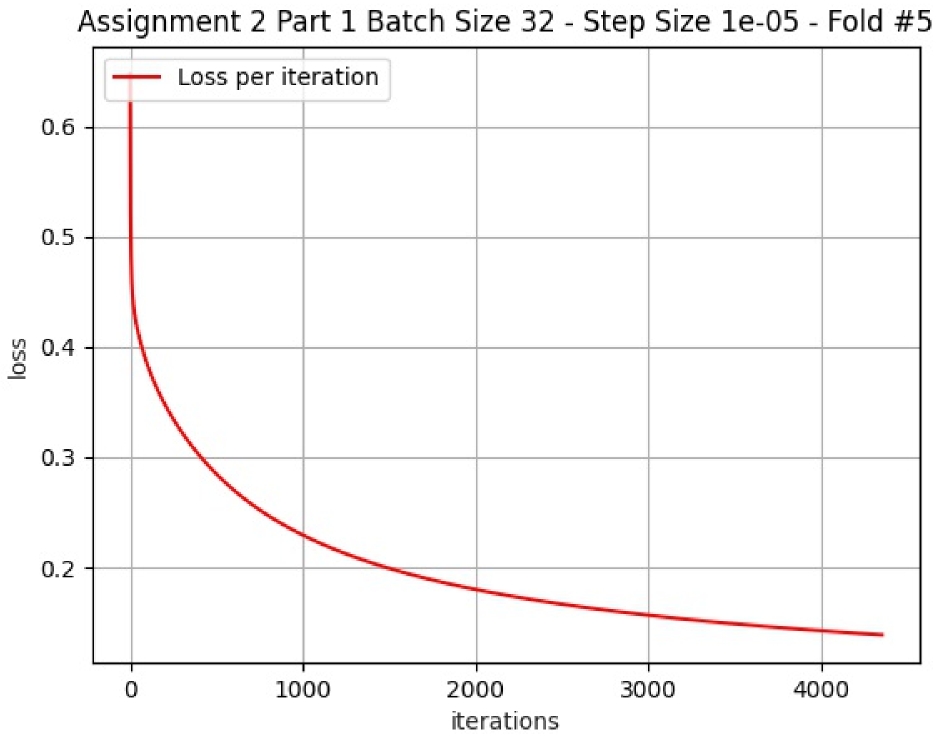
Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big) - Fold #3



Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big) - Fold #4



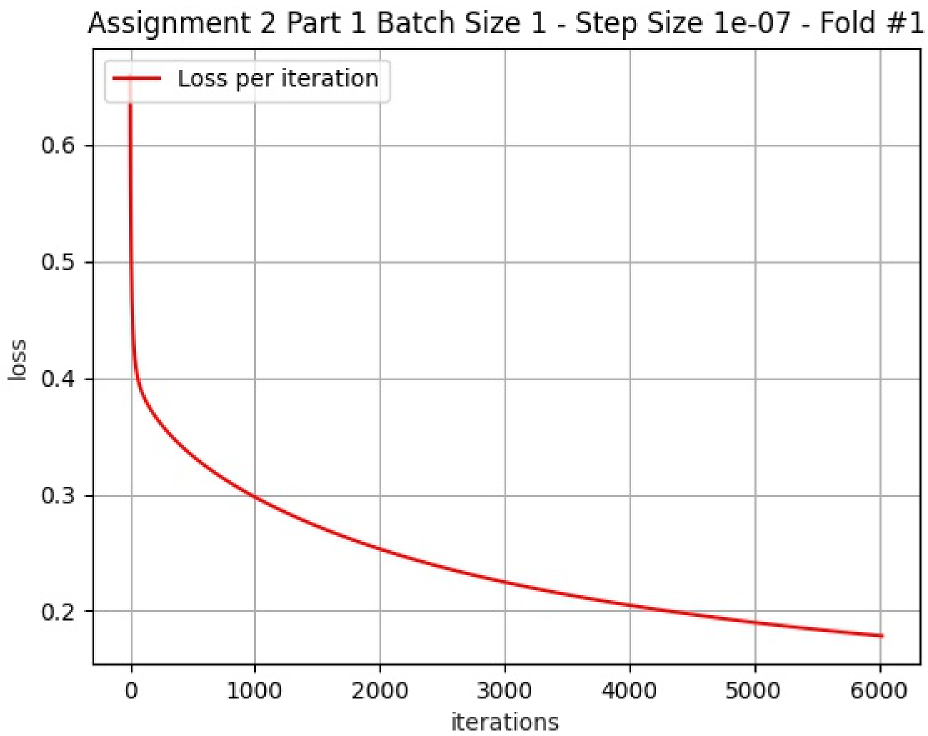
Batch Size 32 (MiniBatch) - Step Size 1e-5 (Big) - Fold #5



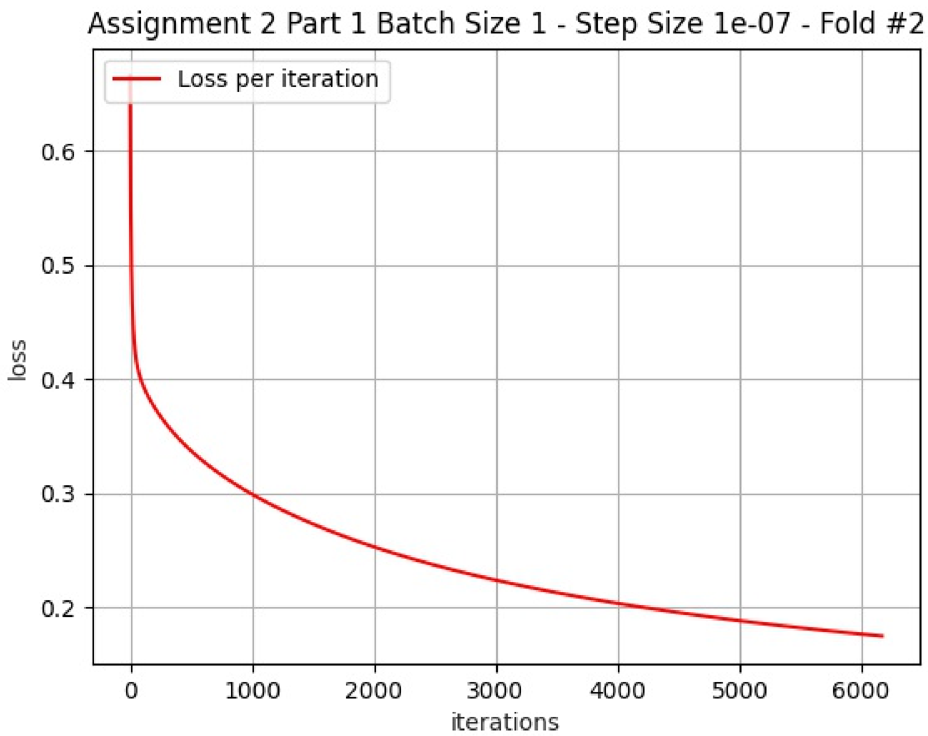
Batch Size inf (SGD) - Step Size 1e-7 (Small)

Average err_train = 0.047789742575712 52
Average err_test = 0.05313534848255619
Stats by Fold:
Fold #1 : iterations = 6024.0 , err_train = 0.049 , err_test = 0.056
Fold #2 : iterations = 6171.0 , err_train = 0.047 , err_test = 0.062
Fold #3 : iterations = 6226.0 , err_train = 0.048 , err_test = 0.046
Fold #4 : iterations = 6004.0 , err_train = 0.047 , err_test = 0.070
Fold #5 : iterations = 6367.0 , err_train = 0.049 , err_test = 0.032

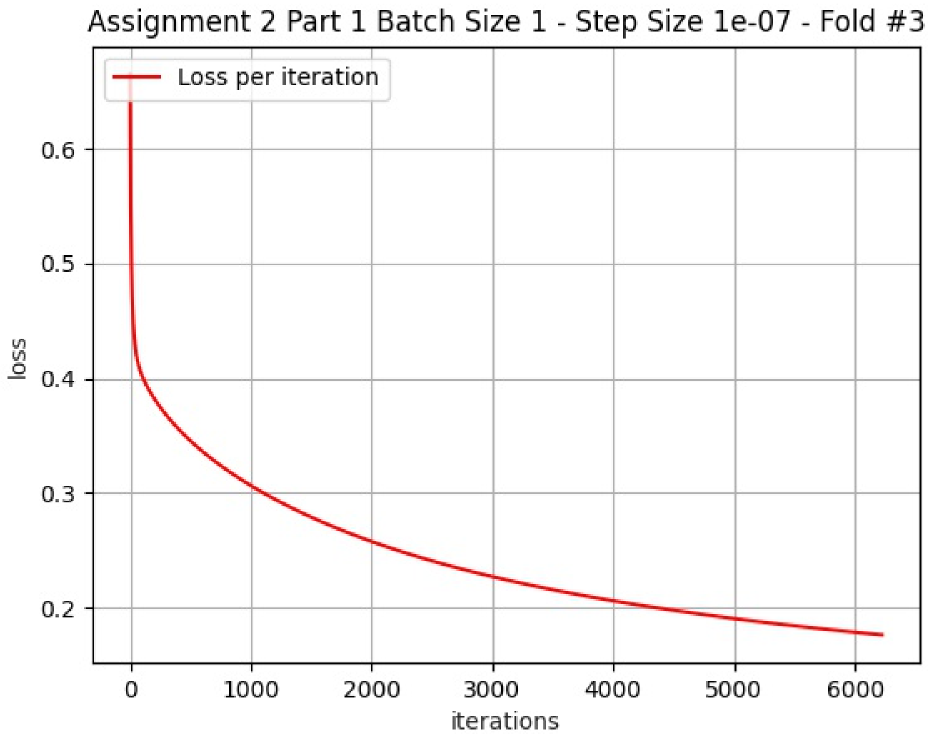
Batch Size inf (SGD) - Step Size 1e-7 (Small) - Fold #1



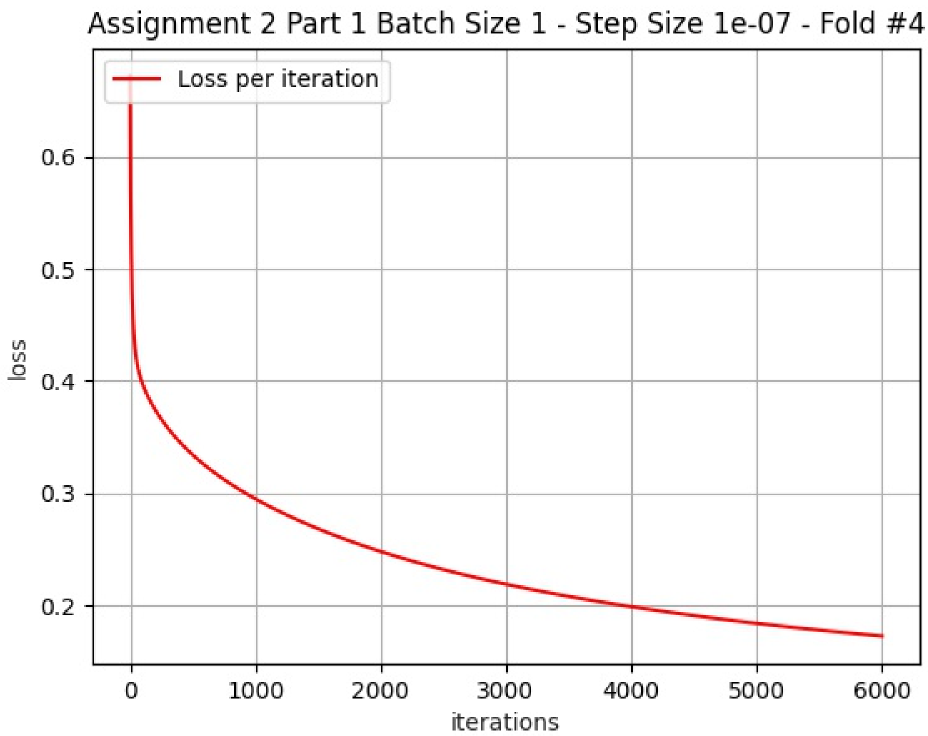
Batch Size inf (SGD) - Step Size 1e-7 (Small) - Fold #2



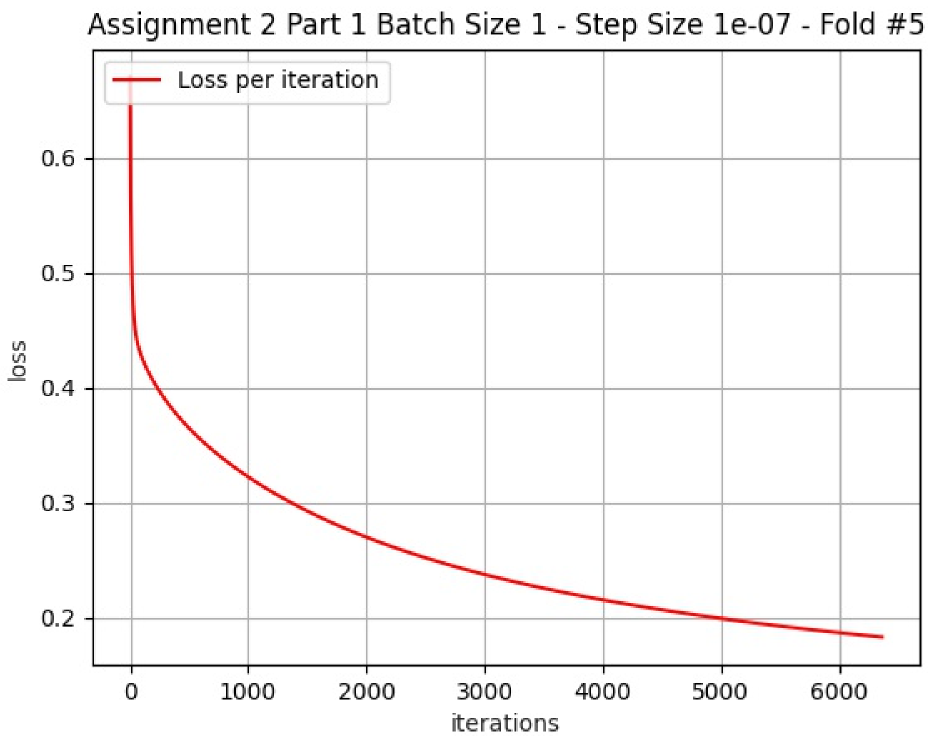
Batch Size inf (SGD) - Step Size 1e-7 (Small) - Fold #3



Batch Size inf (SGD) - Step Size 1e-7 (Small) - Fold #4



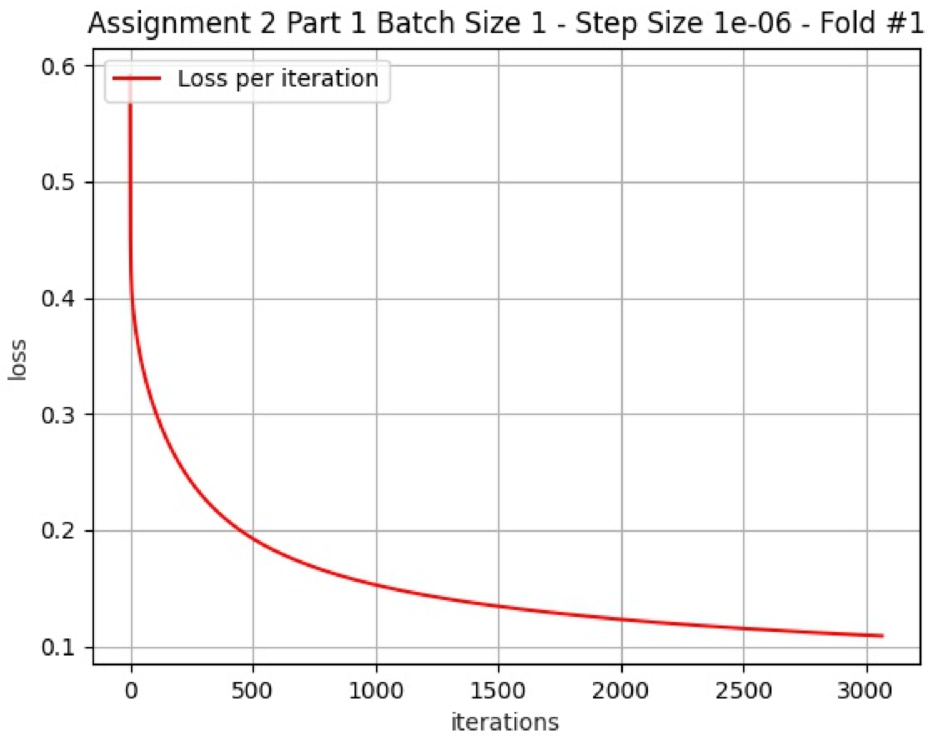
Batch Size inf (SGD) - Step Size 1e-7 (Small) - Fold #5



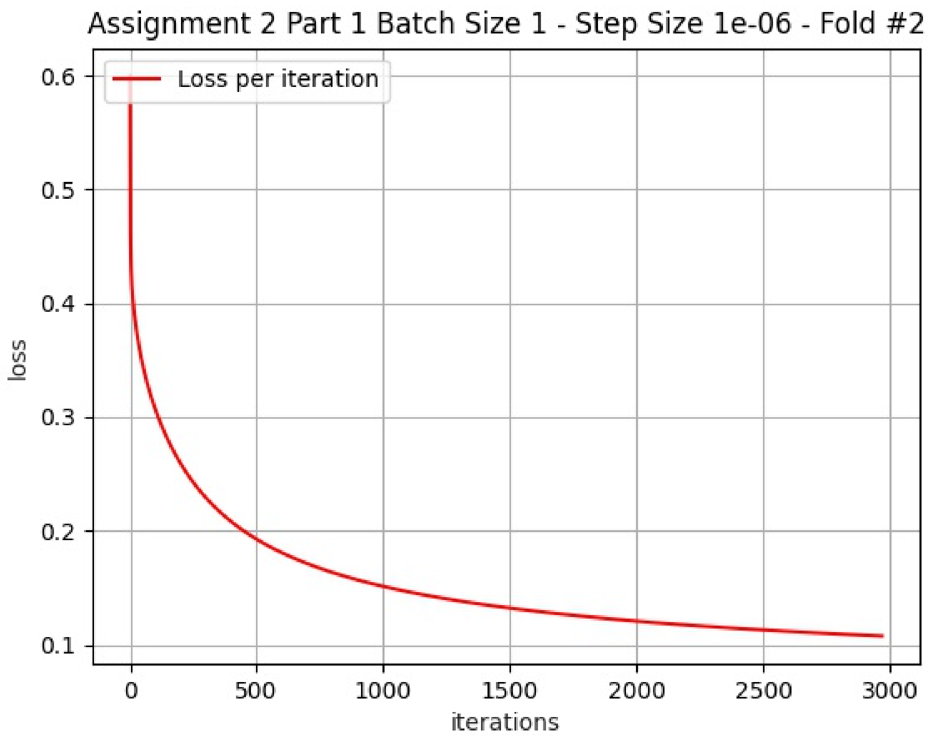
Batch Size inf (SGD) - Step Size 1e-6 (Medium)

Average err_train = 0.028376515438427414
Average err_test = 0.0348464826120811
Stats by Fold:
Fold #1 : iterations = 3066.0 , err_train = 0.029 , err_test = 0.034
Fold #2 : iterations = 2971.0 , err_train = 0.028 , err_test = 0.036
Fold #3 : iterations = 3177.0 , err_train = 0.028 , err_test = 0.033
Fold #4 : iterations = 3174.0 , err_train = 0.027 , err_test = 0.047
Fold #5 : iterations = 3217.0 , err_train = 0.030 , err_test = 0.024

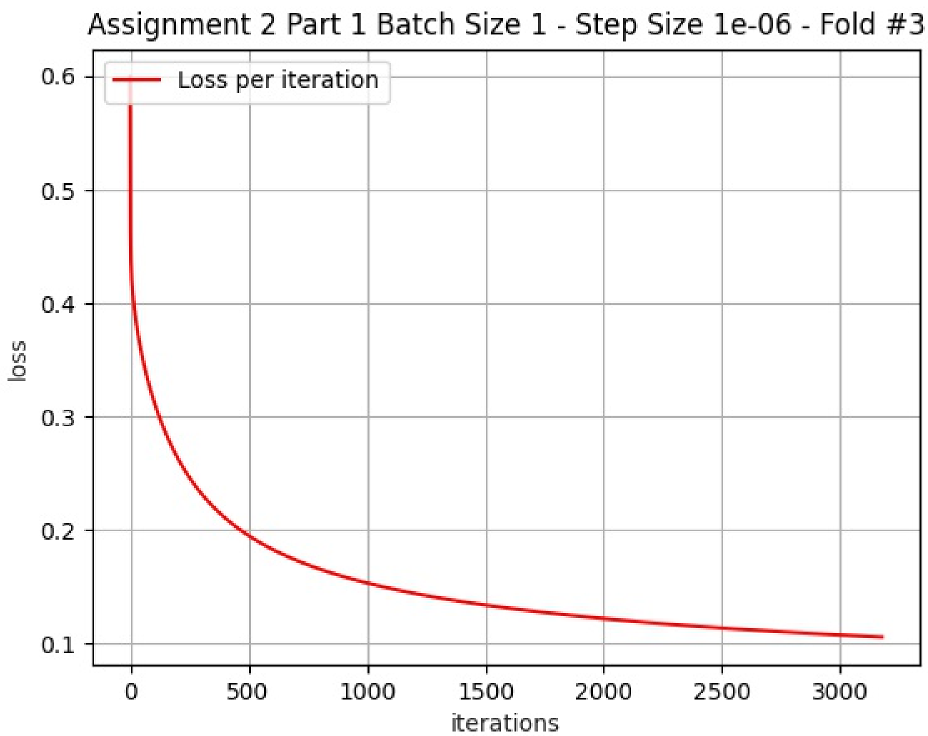
Batch Size inf (SGD) - Step Size 1e-6 (Medium) - Fold #1



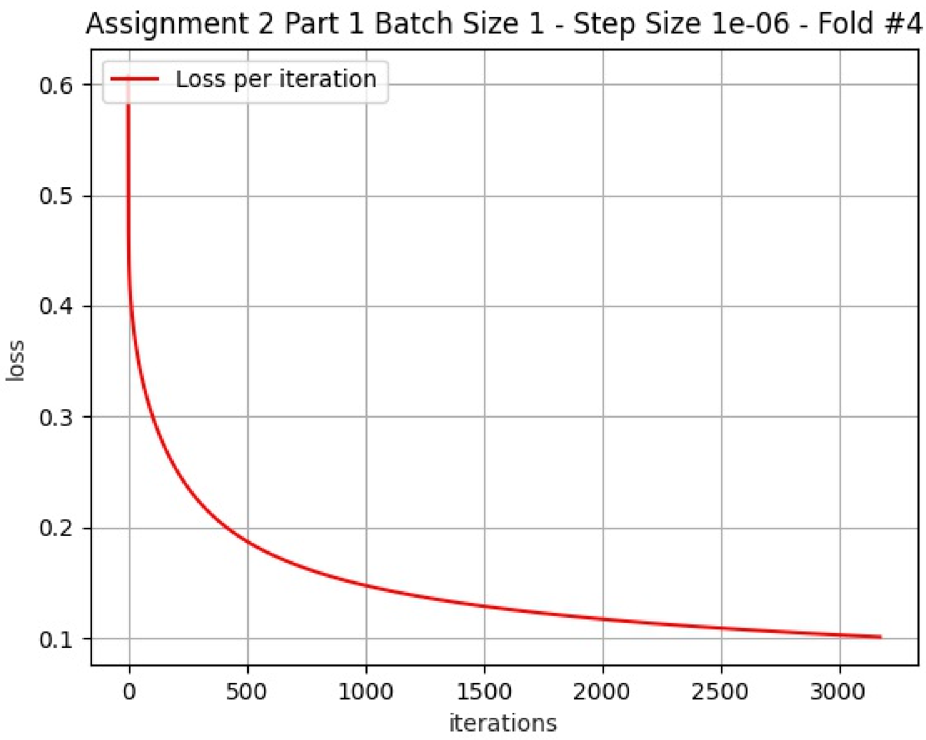
Batch Size inf (SGD) - Step Size 1e-6 (Medium) - Fold #2



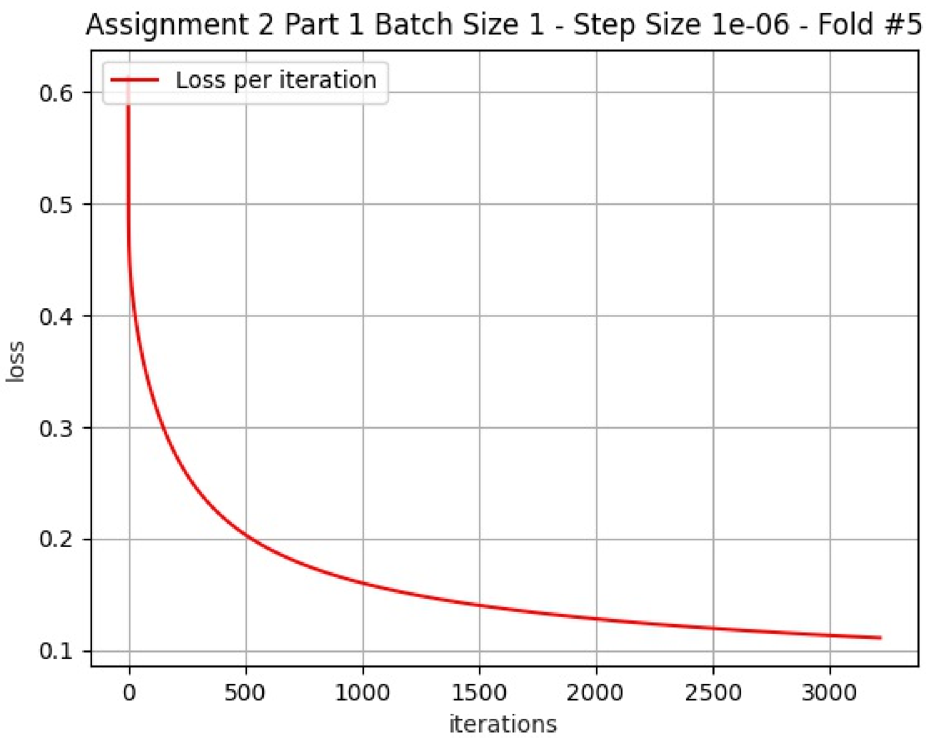
Batch Size inf (SGD) - Step Size 1e-6 (Medium) - Fold #3



Batch Size inf (SGD) - Step Size 1e-6 (Medium) - Fold #4



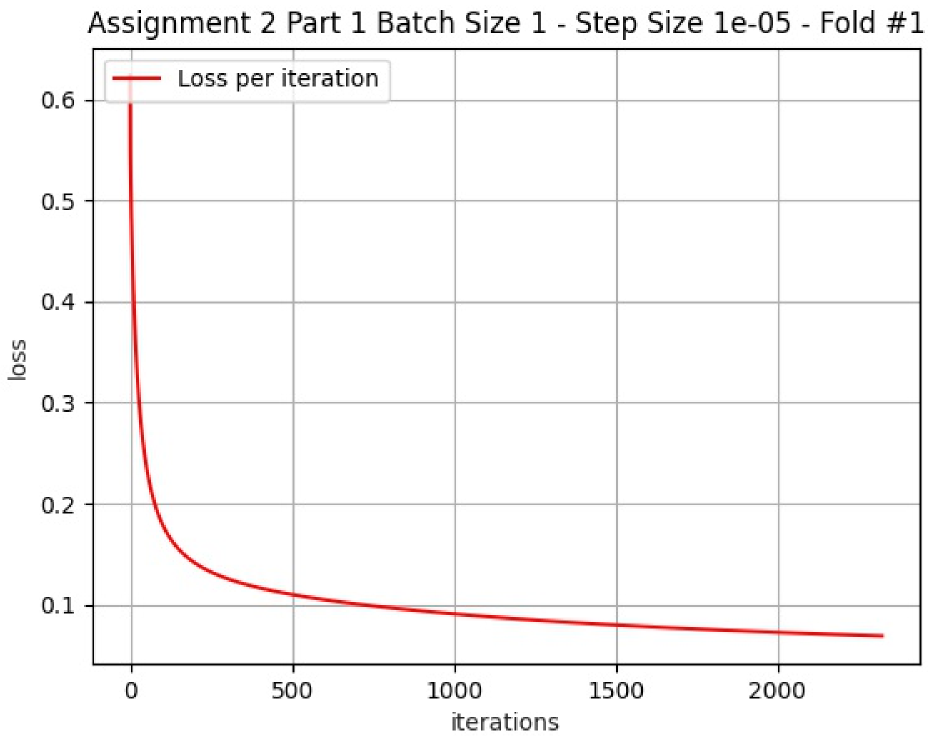
Batch Size inf (SGD) - Step Size 1e-6 (Medium) - Fold #5



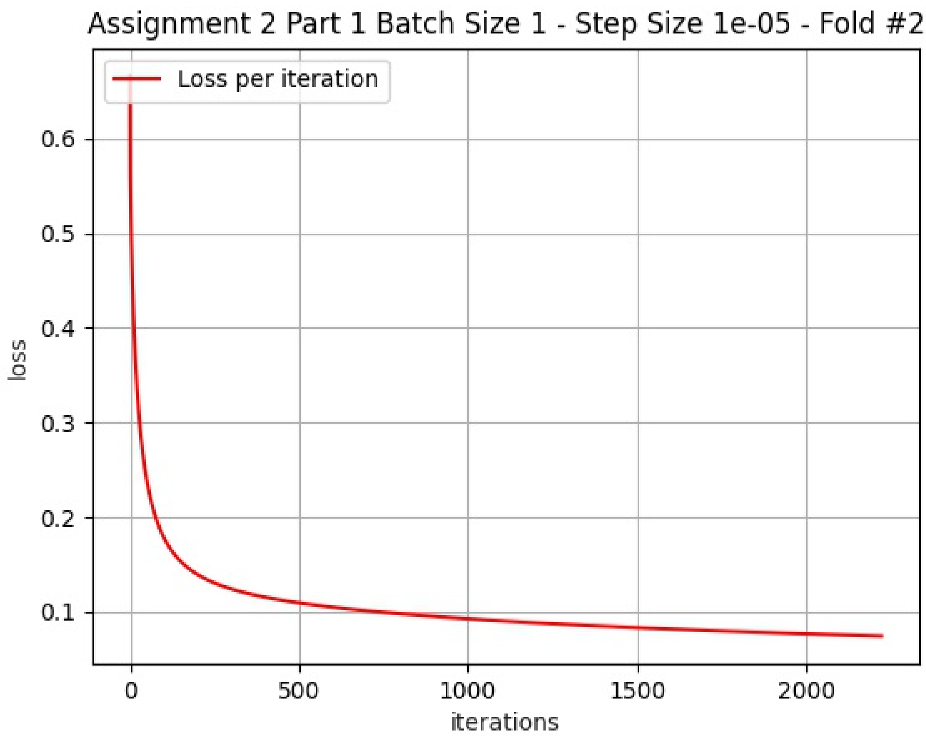
Batch Size inf (SGD) - Step Size 1e-5 (Big)

Average err_train = 0.017028992194655757
Average err_test = 0.026250552778676543
Stats by Fold:
Fold #1 : iterations = 2324.0 , err_train = 0.016 , err_test = 0.030
Fold #2 : iterations = 2225.0 , err_train = 0.019 , err_test = 0.017
Fold #3 : iterations = 2386.0 , err_train = 0.015 , err_test = 0.029
Fold #4 : iterations = 2344.0 , err_train = 0.016 , err_test = 0.039
Fold #5 : iterations = 2342.0 , err_train = 0.018 , err_test = 0.017

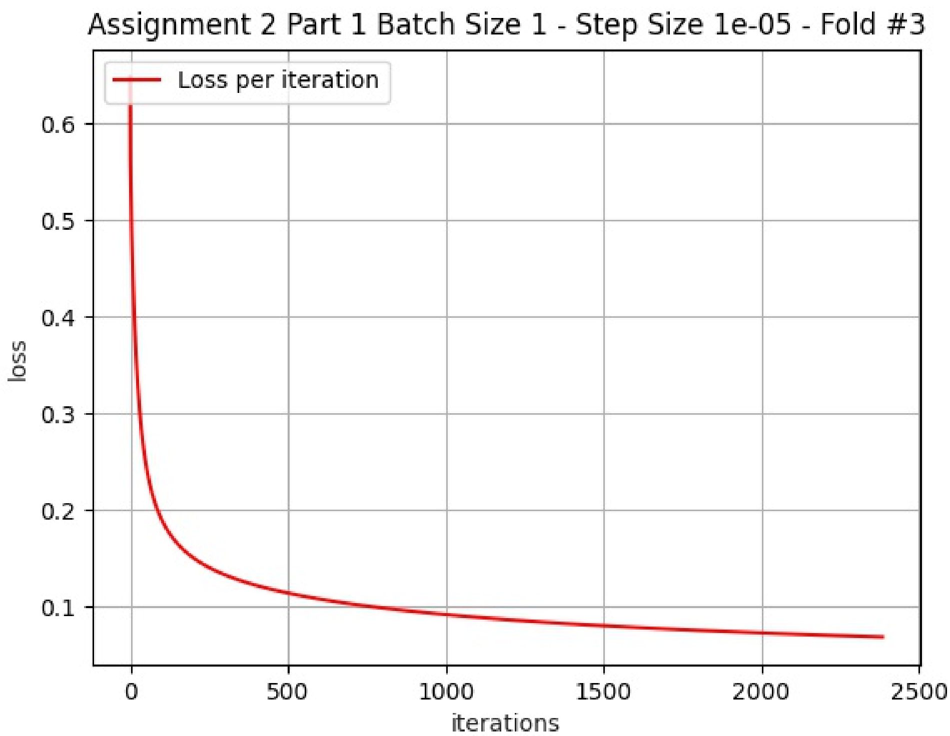
Batch Size inf (SGD) - Step Size 1e-5 (Big) - Fold #1



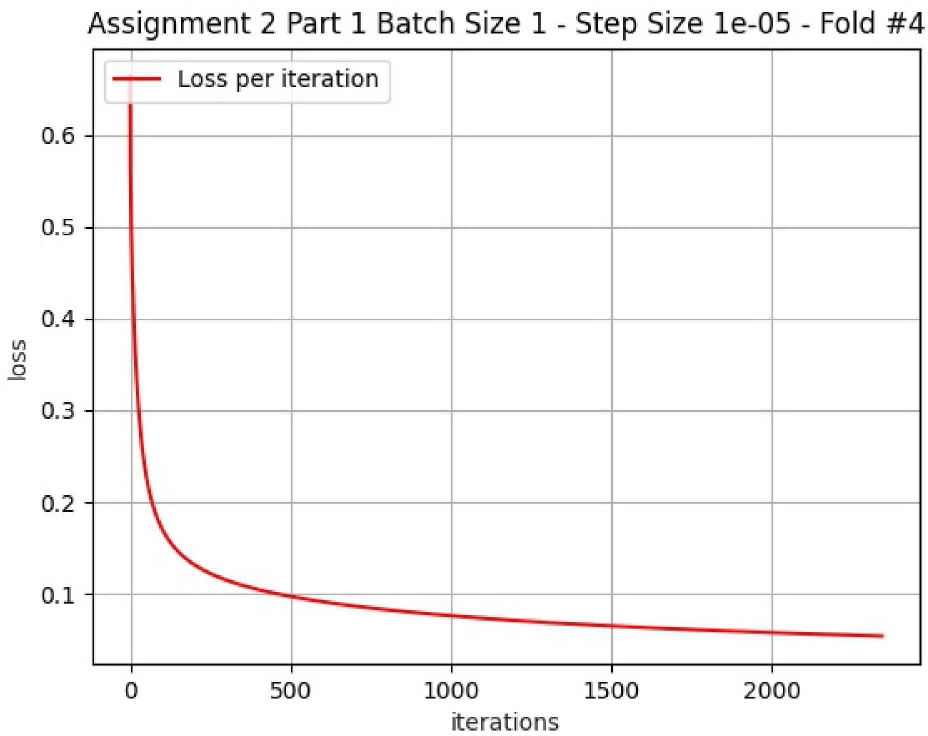
Batch Size inf (SGD) - Step Size 1e-5 (Big) - Fold #2



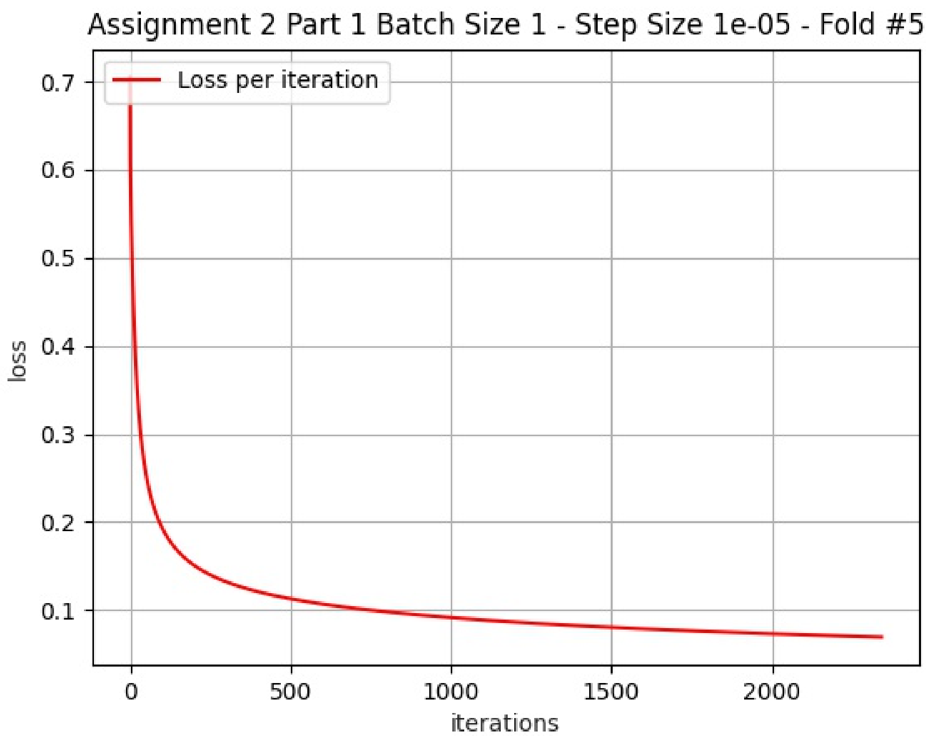
Batch Size inf (SGD) - Step Size 1e-5 (Big) - Fold #3



Batch Size inf (SGD) - Step Size 1e-5 (Big) - Fold #4



Batch Size inf (SGD) - Step Size 1e-5 (Big) - Fold #5



Part 1 Overview

When we check out above results, we see no big difference between each folds and no fold has a very big difference in its test vs train errors. So, it is safe to take average of results over folds and investigate those.

I can't see a pattern regarding avg iterations and step size/batch_size/error. So, I can say that it differs somewhat randomly for different parameters. My guess is that it is dependent on how lucky I are to come across a local/global maxima earlier with the current parameters, since the input includes many (18) features and thus many random-ish local maximas.

The rate of error train and test seems close in each case, thus none of them are overfit.

We can see that both average error train and test are lower in smaller batch sizes or bigger step sizes while avg iterations are not directly affected with those changes.

Not included in results but, batch_size is inversely correlated with training speed as it can't leverage fast matrix multiplication operations with same weights, on smaller batch sizes.

- batch-size inf with default step-size takes 2.5 seconds
- batch-size 32 with default step-size takes 26.5 seconds
- batch-size 1 with default step-size takes 271.1 seconds

When considering the trade-off of better error rates vs training time, mini-batch offers the best fit as its error is only slightly higher than SGD but runs much faster. Likewise, it runs only slightly slower than FGD but its error rate is less.

See below tables for comparing above results more easily.

Batch Size	Step size	Avg Error Train	Avg Error Test	Avg Iterations
inf	1e-07	0.151	0.153	6388.4
inf	1e-06	0.131	0.133	2790.0
inf	1e-05	0.072	0.077	7353.0
32	1e-07	0.130	0.132	2824.2
32	1e-06	0.069	0.073	7410.0
32	1e-05	0.035	0.041	4304.0
1	1e-07	0.048	0.053	6158.4
1	1e-06	0.028	0.035	3121.0
1	1e-05	0.017	0.026	2324.2

Batch Size	Step size	Fold	Error Train	Error Test	Iterations
inf	1e-07	1	0.145	0.178	6473
inf	1e-07	2	0.149	0.155	6485
inf	1e-07	3	0.150	0.153	6358
inf	1e-07	4	0.149	0.150	6576
inf	1e-07	5	0.160	0.128	6050
inf	1e-06	1	0.126	0.159	2724
inf	1e-06	2	0.130	0.136	2825
inf	1e-06	3	0.131	0.135	2678
inf	1e-06	4	0.127	0.138	3038
inf	1e-06	5	0.141	0.098	2685
inf	1e-05	1	0.074	0.087	6927
inf	1e-05	2	0.072	0.090	7126
inf	1e-05	3	0.072	0.067	7552
inf	1e-05	4	0.070	0.096	7162
inf	1e-05	5	0.073	0.045	7998
32	1e-07	1	0.126	0.159	2728
32	1e-07	2	0.129	0.135	2853
32	1e-07	3	0.130	0.134	2694
32	1e-07	4	0.126	0.137	3095
32	1e-07	5	0.139	0.097	2751
32	1e-06	1	0.070	0.084	7020
32	1e-06	2	0.068	0.085	7248
32	1e-06	3	0.068	0.064	7596
32	1e-06	4	0.066	0.092	7228
32	1e-06	5	0.070	0.043	7958
32	1e-05	1	0.036	0.040	4251
32	1e-05	2	0.034	0.045	4287
32	1e-05	3	0.035	0.037	4362
32	1e-05	4	0.033	0.054	4268
32	1e-05	5	0.037	0.027	4352
1	1e-07	1	0.049	0.056	6024
1	1e-07	2	0.047	0.062	6171
1	1e-07	3	0.048	0.046	6226
1	1e-07	4	0.047	0.070	6004
1	1e-07	5	0.049	0.032	6367
1	1e-06	1	0.029	0.034	3066
1	1e-06	2	0.028	0.036	2971
1	1e-06	3	0.028	0.033	3177
1	1e-06	4	0.027	0.047	3174
1	1e-06	5	0.030	0.024	3217
1	1e-05	1	0.016	0.030	2324
1	1e-05	2	0.019	0.017	2225
1	1e-05	3	0.015	0.029	2386
1	1e-05	4	0.016	0.039	2344
1	1e-05	5	0.018	0.017	2342

Part 2

Below is the given input data

Name	GiveBirth	CanFly	LiveInWater	HaveLegs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals
test	yes	no	yes	no	???

When I count mammal vs non-mammal count per class I can obtain below table:

feature	# mammals	# non-mammals
GiveBirth (yes)	6	1
GiveBirth (no)	1	12
CanFly (yes)	1	3
CanFly (no)	6	10
LiveInWater (yes)	2	3
LiveInWater (sometimes)	0	4
LiveInWater (no)	5	6
HaveLegs (yes)	5	9
HaveLegs (no)	2	4

We are asked to guess whether the "test" belongs to "mammals" or "non-mammals" class.

P(mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
P(non-mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)

Apply Bayes Rule:

P(Class|GiveBirth, CanFly, LiveInWater, HaveLegs)
∝
P(GiveBirth, CanFly, LiveInWater, HaveLegs|Class) * P(Class)

We can rewrite then rewrite it as follows:

P(GiveBirth, CanFly, LiveInWater, HaveLegs|Class) * P(Class)
∝
P(GiveBirth|Class) * P(CanFly|Class)
* P(LiveInWater|Class) * P(HaveLegs|Class) * P(Class)

For our case:

P(mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
∝
P(GiveBirth=yes|mammals) * P(CanFly=no|mammals)
* P(LiveInWater=yes|mammals) * P(HaveLegs=no|mammals) * P(mammals)
and
P(non-mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
∝
P(GiveBirth=yes|non-mammals) * P(CanFly=no|non-mammals)
* P(LiveInWater=yes|non-mammals) * P(HaveLegs=no|non-mammals) * P(non-mammals)

Using the frequency table we can compute all these:

P(GiveBirth=yes|mammals) = 6/7
P(CanFly=no|mammals) = 6/7
P(LiveInWater=yes|mammals) = 2/7
P(HaveLegs=no|mammals) = 2/7
P(mammals) = 7/20

P(GiveBirth=yes|non-mammals) = 1/13
P(CanFly=no|non-mammals) = 10/13
P(LiveInWater=yes|non-mammals) = 3/13
P(HaveLegs=no|non-mammals) = 4/13
P(non-mammals) = 13/20

When we multiply these values as in previous formula we obtain below results:

P(mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
∝
6/7 * 6/7 * 2/7 * 2/7 * 7/20 = 36/1715 ~ 0.0209912536

and

P(non-mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
∝
1/13 * 10/13 * 3/13 * 4/13 * 13/20 = 30/10985 ~ 0.00273099681

Now we can compare these two results to make our decision:

36/1715 > 30/10985
0.0209912536 > 0.00273099681
P(mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)
>
P(non-mammals | GiveBirth=yes, CanFly=no, LiveInWater=yes, HaveLegs=no)

Thus, I would guess that "test" belongs to "mammals" class.