EMORY
UNIVERSITY

Department of Computer Science & Informatics

Masoud Nateghi

CS534 HW1

Professor Joyce Ho

1. Show that the ridge regression estimates can be obtained by ordinary least squares regression on an augmented data set. We augment the centered matrix X with k additional rows $\sqrt{\lambda}I_{k\times k}$ and augment y with k zeros. The idea is that by introducing artificial data having response value zero, the fitting procedure is forced to shrink the coefficients toward zero.

Now we can consider the new augmented variables $\tilde{X}$ and $\tilde{y}$ to be of the following form:

$$\tilde{X} = \begin{bmatrix} X \\ \sqrt{\lambda}I_{k\times k} \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

We know that for an OLS regression problem, we can find the coefficient vector $\tilde{\beta}$ from regular data matrix X and target variable y using the following formula:

$$\tilde{\beta} = (X^T X)^{-1} X^T y$$

We can apply this formula to our new set of augmented variables:

$$\tilde{\beta} = \left(\tilde{X}^T \tilde{X}\right)^{-1} \tilde{X}^T \tilde{y}$$

The terms $\tilde{X}^T\tilde{X}$ and $\tilde{X}^T\tilde{y}$ could be simplified:

$$\tilde{X}^T\tilde{X} = X^T X + \lambda I_{p\times p}$$

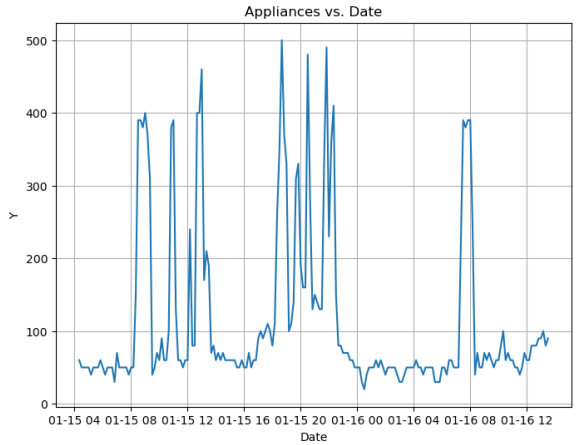
$$\tilde{X}^T\tilde{y} = X^T y$$

$$\tilde{\beta} = \left(X^T X + \lambda I_{p\times p}\right)^{-1} X^T y$$

Which is exactly the solution for ridge regression.

2. (a) How did you preprocess the data? Explain your reasoning for using this pre-processing.

We can specifically use ‘date’ column contents, which in my opinion are super important. For instance, people usually get up at 8 a.m. and eat breakfast, so they will use more appliances than that they use at 3 a.m. We can use from date time as a feature. In my implementation, I used 2-hour intervals (12 categorical features) for the task of regression. As this figure shows, we have some peaks in certain time intervals which can be used to predict appliances. Another important fact is that, this time intervals may vary depending on the day of the week (7 categorical features). However, this feature might be useless but the good point is that in regression models, the coefficient for this kind of variables go to zero. We will also separate the output Appliances form the dataset and put the features matrices in trainX, valX, and testX. Also, we store target outputs (Appliances) in trainY, valY, and testY. This are all implemented in **feature_extraction** function.



preprocess function description:

I first checked for any nan values to drop rows with nan values, however the dataset did not have any. Finally, we tend to apply some normalization to our features (not categorical features) as always!

2. (e) Report (using a table) the RMSE and R2 between 2c and 2d on the energydata. How do the performances compare and what do the numbers suggest?

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_linear1	94.02	0.255	94.58	0.062	89.93	0.021
eval_linear2	95.32	0.234	**86.12**	**0.223**	**83.76**	**0.150**

From the table above we can conclude that, **by using validation data, we can achieve better performance**. However, we should remember that, most of the time we do not have access to the test data, and all we have are a bunch of data which we can split them into train and validation data. We first select our model by training different models on the train and validation data. We use validation data as a data which model has not seen yet to monitor the performance of the model and avoid overfitting. At the end of the day, **when we have selected our best model from the given data, it would be a good idea to train the model on both train and validation data**.

2. (h) Report (using a table) the RMSE and R2 for training, validation, and test for all the different (λ) values you tried. What would be the optimal parameter you would select based on the validation data performance?

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_ridge1 (λ = 0)	94.02	0.254	94.58	0.062	89.93	0.020
eval_ridge1 (λ = 250)	94.68	0.244	92.42	0.104	87.93	0.063
eval_ridge1 (λ = 500)	95.44	0.232	92.15	0.109	87.50	0.072
eval_ridge1 (λ = 750)	**96.09**	**0.221**	**92.09**	**0.110**	**87.31**	**0.076**
eval_ridge1 (λ = 1000)	96.63	0.213	92.10	0.110	87.22	0.078

The model is selected usually based on the performance of the model on the validation data. Here, I chose valid-rmse as the metric to select the best model. It can be seen that for λ = 750, we have the best performance. Again, we can run a code to sweep λ values from 0 to 1000 with 1001 points (or even search around the optimal λ more precisely) to find the optimal λ value automatically. This will result in $\lambda_{opt} = 822$. We could make our search narrower by searching over $\boldsymbol{\lambda^*_{ridge} = 822}$.

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_lasso1 (λ = 0)	94.02	0.254	94.49	0.064	89.87	0.021
eval_lasso1 (λ = 1)	95.60	0.229	93.66	0.080	88.24	0.056
eval_lasso1 (**λ = 2**)	**97.93**	**0.191**	**92.60**	**0.100**	**87.48**	**0.073**
eval_lasso1 (λ = 3)	100.06	0.156	92.83	0.096	87.80	0.066
eval_lasso1 (λ = 4)	101.33	0.134	93.42	0.085	88.14	0.059

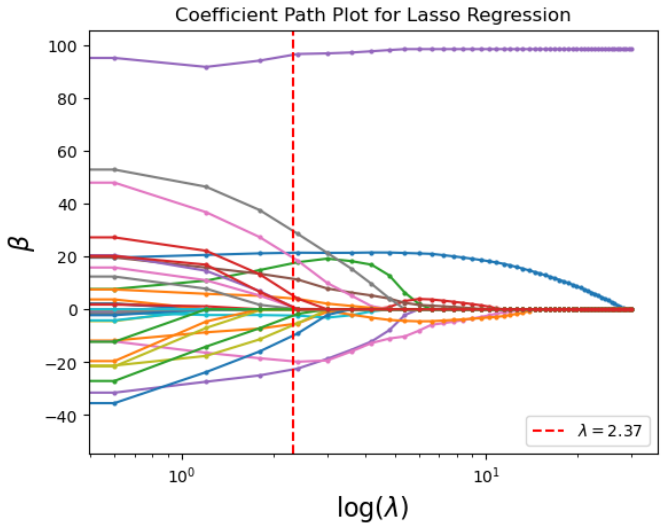
The model is selected usually based on the performance of the model on the validation data. Here, I chose valid-rmse as the metric to select the best model. It can be seen that for λ = 2, we have the best performance. Again, we can run a code to sweep λ values from 0 to 4 with 1001 points to find the optimal λ value automatically. This will result in $\lambda_{opt} = 2.32$. We could make our search narrower by searching over $\boldsymbol{\lambda^*_{lasso} = 2.32}$.

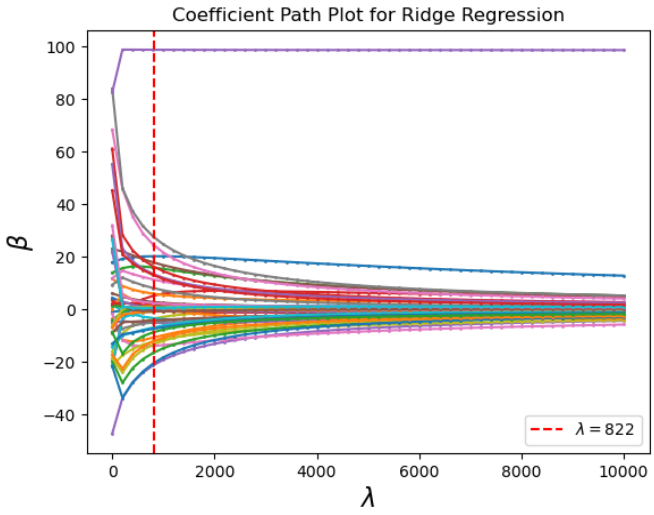
2. (j) Use the optimal regularization parameter from 2h and report the RMSE and R2 on the training set, validation set, and test set for the functions you wrote on 2i? How does this compare to the results from 2h? What do the numbers suggest?

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_ridge1(λ = 822)	96.25	0.219	92.09	0.110	87.28	0.077
eval_ridge2(λ = 822)	96.74	0.211	87.03	0.205	**83.64**	**0.152**
eval_lasso1(λ = 2.32)	98.69	0.179	92.51	0.102	87.42	0.074
eval_lasso2(λ = 2.32)	100.57	0.147	90.16	0.147	**85.70**	**0.110**

For each model, we can see that **by using validation data, we can achieve better performance** (valid-rmse as metric). At the end of the day, when we have selected our best model from the given data, it would be **a good idea to train the model on both train and validation data**. Also, it can be seen that **ridge regression performs slightly better** on the test data.

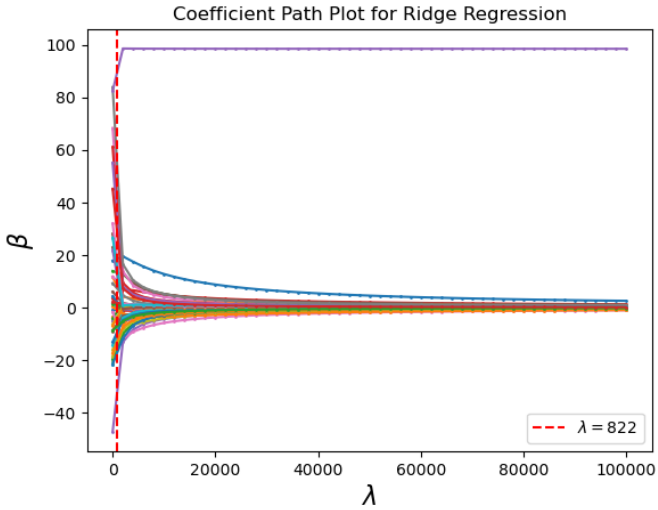
2. (k) Generate the coefficient path plots (regularization value vs. coefficient value) for both ridge and lasso. Also, note (line or point or star) where the optimal regularization parameters from 2h are on their respective plots. Make sure that your plots encompass all the expected behavior (coefficients should shrink towards 0).





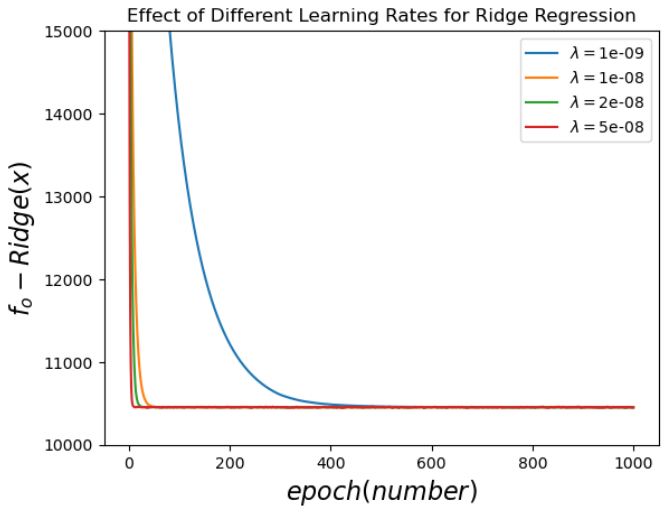
2. (l) What are 3 observations you can draw from looking at the coefficient path plots, and the metrics? This should be different from your observations from 2e, 2h, and

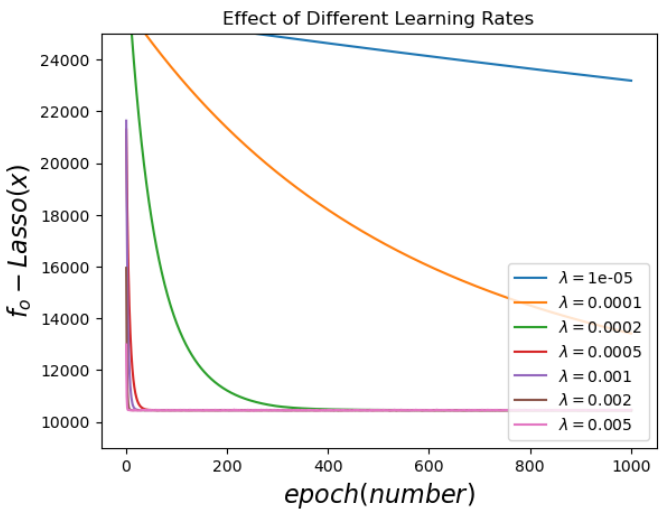
The purple line is corresponded to the bias wight which is not penalized; so, the weight remains constant even for bigger values of λ. From both plots it can bee seen that the coefficients shrink to zero as λ gets bigger and bigger. Also. the blue line indicates the coefficient for 'lights' which becomes zero later than other variables. This is because the target value has a stronger correlation with this feature than other features. Moreover, ridge regression needs bigger λ values to make the coefficients zero. This becomes clear when we consider the shape plot of L_1 and L_2 norms and recall that lasso is accompanied with sparser coefficients. In the first sight, it might not be obvious for the viewer that in ridge regression coefficients get closer to zero; however, this will become more clear if we run the simulation for even bigger λ values. Below figure illustrates this concept:



3. (f) For the optimal regularization parameters from ridge (λ_{ridge}) and lasso (λ_{lasso}) from 2h, and $\alpha = \frac{1}{2}$, what are good learning rates for the dataset? Justify the selection by trying various learning rates and illustrating the objective value ($f_o(x)$) on a graph for a range of epochs (one epoch = one pass through the training data). For the chosen learning rate you identified, what are the RMSE and R2 for the elastic net model trained on the entire training set on the training, validation, and test sets?

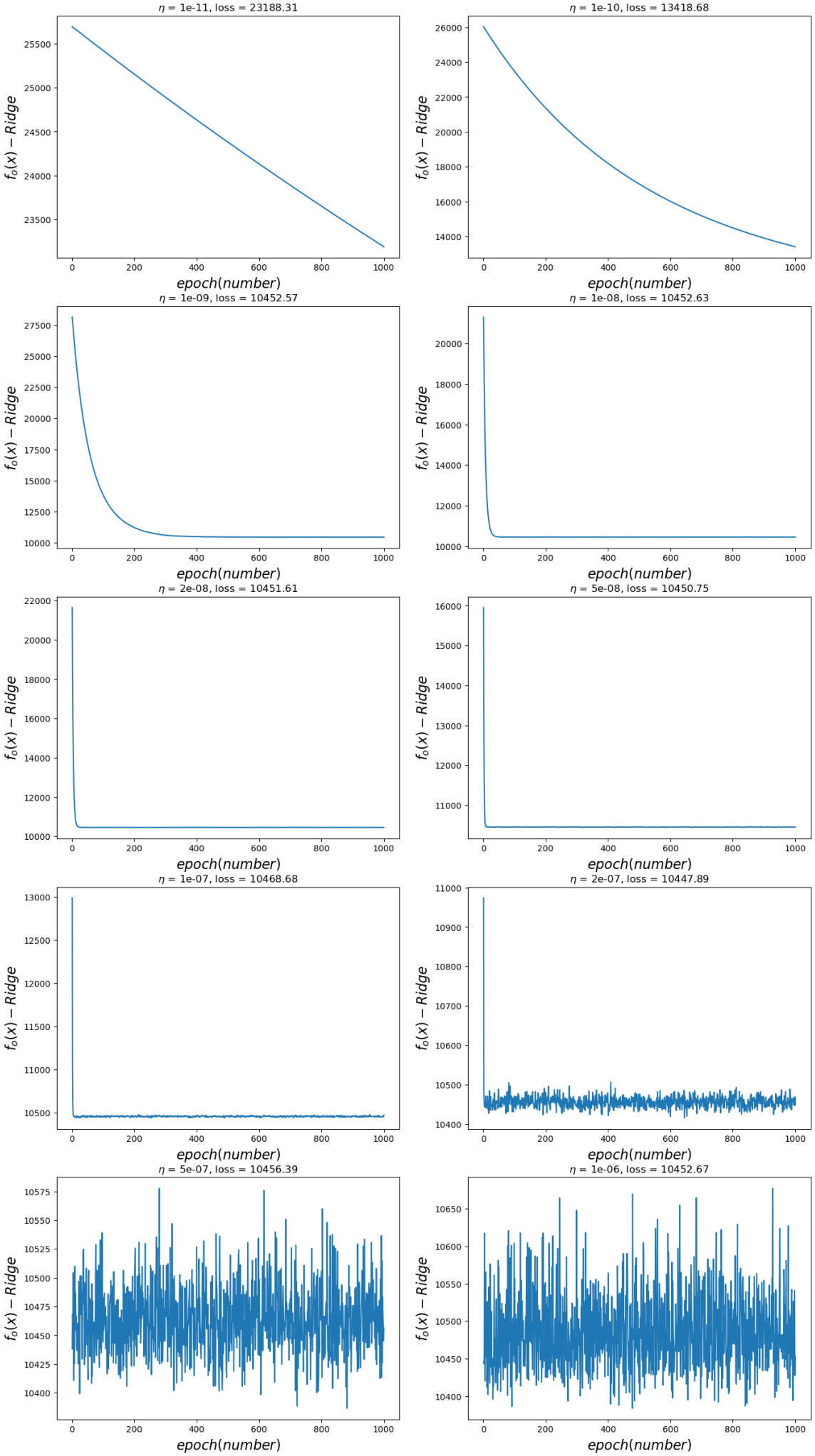
We run the simulation two times for tuning the learning rate. This implies the importance of determining a proper learning rate for the problem we are dealing with as we will see different learning rates in the order of magnitude, when we run the simulation for (alpha = 0.5, el = 822) which is the optimal parameter for ridge regression and (alpha = 0.5, el = 2.32) that is the optimal parameter for lasso regression. To find the optimal step size, we can use a portion of the data as SGD, in theory, is not sensitive to the dataset. Thus, we can subsample a reasonable percentage of data to tune the learning rate and we used 20% of the training set (random selection) to do this task. If we choose small learning rate, the coefficient changes are really minor, and it takes a lot of epochs to reach to global minima. However, if we take very large step size, the coefficients either oscillate around minimum or even the algorithm might diverge. So, our ideal step size is large enough to reach the coefficients to minimum as fast as possible. And, this step size is not that large to cause any oscillations or divergence. If we look at the figures below and figures on the next page, we can guess that for ridge and lasso regression a good choice for learning rate can be $\eta_{ridge} = 5 \times 10^{-8}$ and $\eta_{lasso} = 5 \times 10^{-3}$ respectively.

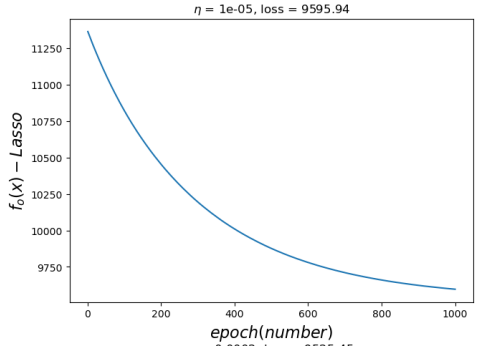


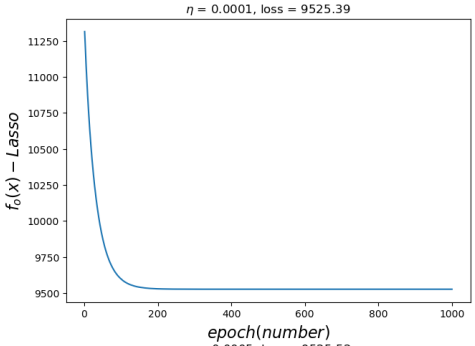


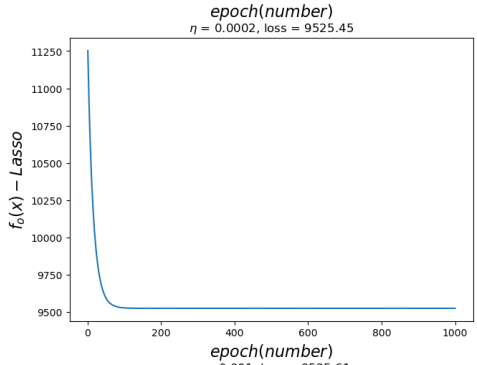
	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_elastic ($\lambda = 822, \alpha = 0.5, \eta = 5 \times 10^{-8}$)	146.74	-0.814	137.44	-0.98	132.74	-1.13
eval_elastic ($\lambda = 2.37, \alpha = 0.5, \eta = 5 \times 10^{-8}$)	123.55	-0.286	114.20	-0.367	108.56	-0.427

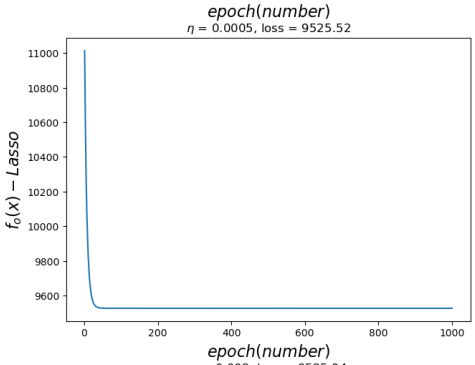
It is obvious that the current setting for the parameters is not ideal for us as the errors are not desired for us. However, we could find the optimal learning rates for each model. It is clear that by introducing regularization terms, the error/bias increases in order to sacrifice bias for variance and to have a more generalized model which does not overfit the data.

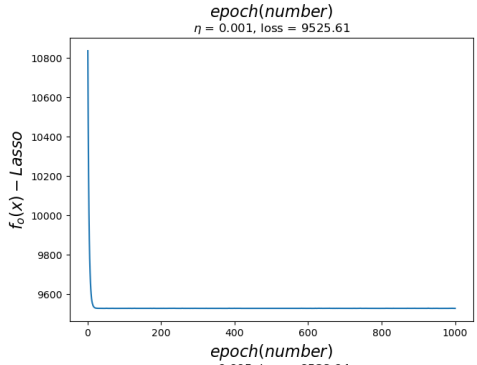

η = 1e-11, loss = 23188.31
η = 1e-10, loss = 13418.68
η = 1e-09, loss = 10452.57
η = 1e-08, loss = 10452.63
η = 2e-08, loss = 10451.61
η = 5e-08, loss = 10450.75
η = 1e-07, loss = 10468.68
η = 2e-07, loss = 10447.89
η = 5e-07, loss = 10456.39
η = 1e-06, loss = 10452.67
$f_o(x) - Ridge$
$epoch(number)$

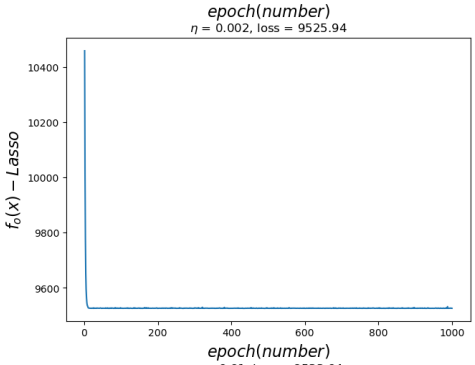

η = 1e-05, loss = 9595.94
$f_o(x) - Lasso$
epoch(number)

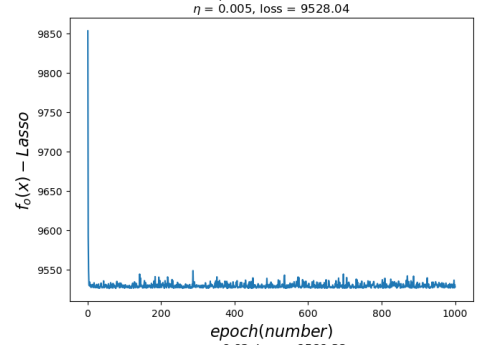

η = 0.0001, loss = 9525.39
$f_o(x) - Lasso$
epoch(number)

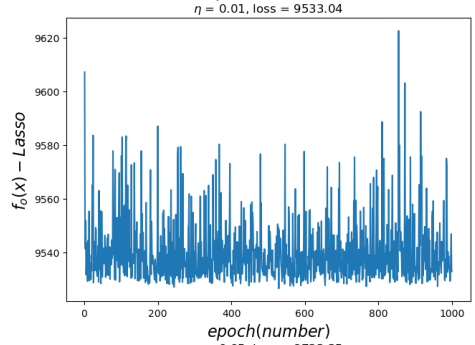

η = 0.0002, loss = 9525.45
$f_o(x) - Lasso$
epoch(number)

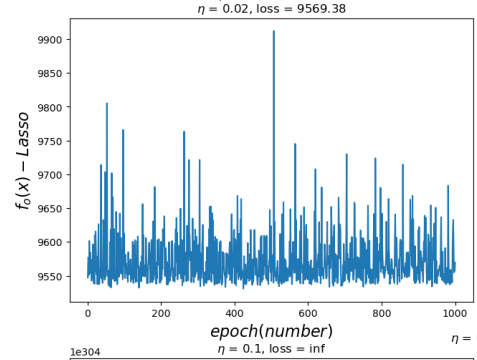

η = 0.0005, loss = 9525.52
$f_o(x) - Lasso$
epoch(number)

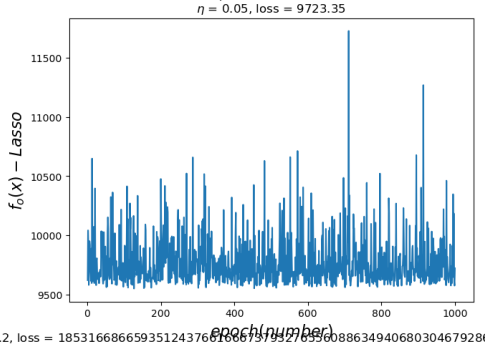

η = 0.001, loss = 9525.61
$f_o(x) - Lasso$
epoch(number)

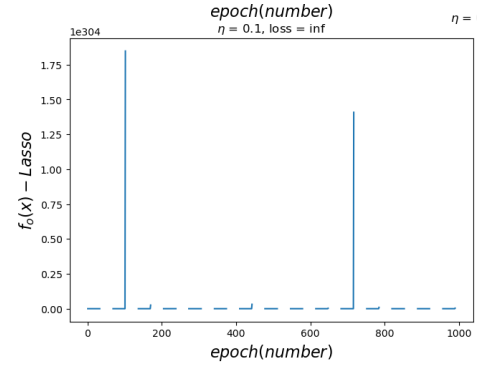

η = 0.002, loss = 9525.94
$f_o(x) - Lasso$
epoch(number)

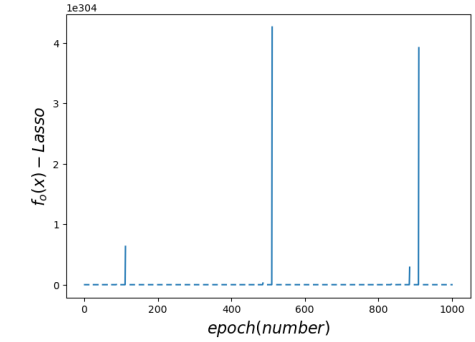

η = 0.005, loss = 9528.04
$f_o(x) - Lasso$
epoch(number)


η = 0.01, loss = 9533.04
$f_o(x) - Lasso$
epoch(number)


η = 0.02, loss = 9569.38
$f_o(x) - Lasso$
epoch(number)


η = 0.05, loss = 9723.35
$f_o(x) - Lasso$
epoch(number)


η = 0.1, loss = inf
1e304
$f_o(x) - Lasso$
epoch(number)


η = 0.2, loss = 185316686659351243766186679179327653608863494068030467928674533376 0.00
1e304
$f_o(x) - Lasso$
epoch(number)

3. (g) Using the learning rate from the previous part, train elastic net (using only training data) for different values of α (it should encompass the entire range and include (α = 0, 1). Report the RMSE and R2 for the models on training, validation, and test set.

For the ridge model we have:

	train-rmse	train-r2	val-rmse	val-r2	test-rmse	test-r2	alpha
0	100.575296	0.147475	93.940832	0.074920	88.214554	0.057660	0.0
1	146.289448	-0.803643	136.971213	-0.966660	132.255887	-1.118151	0.1
2	146.573397	-0.810651	137.263694	-0.975068	132.557466	-1.127822	0.2
3	146.669498	-0.813026	137.362725	-0.977919	132.659393	-1.131096	0.3
4	146.715991	-0.814176	137.410396	-0.979292	132.708838	-1.132685	0.4
5	146.745884	-0.814915	137.441926	-0.980200	132.741100	-1.133722	0.5
6	146.765458	-0.815399	137.461912	-0.980776	132.761604	-1.134381	0.6
7	146.778624	-0.815725	137.475831	-0.981177	132.775664	-1.134833	0.7
8	146.788181	-0.815962	137.484915	-0.981439	132.785511	-1.135150	0.8
9	146.795788	-0.816150	137.492842	-0.981668	132.793335	-1.135401	0.9
10	146.803867	-0.816350	137.501046	-0.981904	132.802491	-1.135696	1.0

And here is the result for the lasso regression:

	train-rmse	train-r2	val-rmse	val-r2	test-rmse	test-r2	alpha
0	95.606991	0.229622	93.826874	0.077163	88.464601	0.052311	0.0
1	104.942158	0.071837	96.808068	0.017588	91.136617	-0.005803	0.1
2	111.495931	-0.047713	102.881454	-0.109545	97.195774	-0.143989	0.2
3	116.555199	-0.144953	107.230751	-0.205339	101.542216	-0.248591	0.3
4	120.279730	-0.219297	110.787865	-0.286634	105.084438	-0.337223	0.4
5	123.487513	-0.285200	114.106147	-0.364862	108.570412	-0.427414	0.5
6	125.728827	-0.332276	116.236847	-0.416310	110.788566	-0.486336	0.6
7	127.798259	-0.376494	118.023133	-0.460175	112.698129	-0.538014	0.7
8	129.575899	-0.415054	120.089102	-0.511743	114.799716	-0.595911	0.8
9	130.867985	-0.443415	121.466390	-0.546617	116.250262	-0.636496	0.9
10	131.942039	-0.467205	122.339625	-0.568935	117.098285	-0.660459	1.0

3. (h) Based on the results from (c) and 2(a) and 2(c), what conclusions can you draw in terms of RMSE and R2? Which model is the best? Also, discuss the differences between the SGD-variants of Ridge and LASSO and the standard implementations (Problem 2).

Our criteria to compare models, is the performance of the models on validation data. Because we normally do not have access to test data, and all we have is validation data. We know that for $\alpha = 1$ $f_o(x)$ will become ridge regression. On the other hand, for $\alpha = 0$ the equation will become the formula for lasso regression. Thus, to compare direct algebraic formula in Q2 with this variant of SGD (Proximal Gradient Descent) in Q3, we can compare the metrics in the first table (ridgre regression) for $\alpha = 1$ and the ones in the second table (lasso regression) for $\alpha = 0$ with the results in part 2j.

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_ridge1(λ = 822)	96.25	0.219	**92.09**	**0.110**	**87.28**	**0.077**
eval_elastic(λ=822, α=1)	146.80	-0.816	137.50	-0.981	132.80	-1.135
eval_lasso1(λ = 2.32)	98.69	0.179	**92.51**	**0.102**	**87.42**	**0.074**
eval_elastic(λ=2.32, α=0)	95.60	0.229	93.82	0.077	88.46	0.052

We can see that the functions in Q2 have better metrics for validation data, and this is what we exactly expected. Because we know that we are dealing with a convex problem that has a closed form solution. As a result, this method is superior to SGD and its variants, because it finds the global minima in one step. However, SGD finds the minimum stochastically, and even there is no guarantee that SGD converges to the global minimum. Amongst all these models, eval_ridge1(λ = 822) has the best performance, and then eval_lasso1(λ = 2.32) performs better with respect to the other methods that use SGD.

3. (i) What are the final coefficients that yield the best elastic net model on the test data? Compare these with the final coefficients for the best-performing model on the validation dataset. Are there noticeable differences? If so, discuss the differences with respect to the impact on the performance.

	train-rmse	train-r2	valid-rmse	valid-r2	test-rmse	test-r2
eval_elastic (λ=822, α=0)	100.57	0.147	93.94	0.074	88.21	0.057
eval_elastic (λ=2.32, α=0)	95.60	0.229	93.82	0.077	88.46	0.052

We evaluated the performance of different models in parts (f) and (g) of the Q3 and we can see a summary of each model's performance in the table above. It is obvious that on the validation dataset, elastic regression with parameters (λ=2.32, α=0) works better, and for the test dataset elastic regression with parameters (λ=2.32, α=0) performs better.

```
[[0.82938678 0.33691016 0.72388827 0.13707166 0.38862308 0.3459486
  0.04988763 0.02111274 0.3523566  0.30264892 0.50673209 0.18277183
  0.70110948 0.00789241 0.60378569 0.78594019 0.41463361 0.01666424
  0.0876099  0.09465056 0.45998475 0.29761923 0.13858926 0.0161371
  0.9318902  0.3433237  0.83880355 0.83992321 0.28301767 0.69692101
  0.06781023 0.13098999 0.9087982  0.11415305 0.74458531 0.29616936
  0.95232216 0.02128931 0.13244598 0.7879304  0.19975062 0.55323623
  0.40424809 0.51838845 0.71240419]]
```

```
[[ 2.03913236e+01 -9.22861242e+00  9.57942537e+00 -8.88709571e-03
  -2.83369943e+01  1.67871752e+01  4.04400205e+01 -4.34039963e-02
  -5.91386997e-02 -2.01263120e+00  1.47240648e-02  6.33272419e+00
   1.66149430e-02 -4.93674406e-02 -6.91454134e-02  3.91969726e-02
  -1.67171864e+01 -4.84527290e-01 -1.96931708e+01 -1.22605916e-02
   3.20445690e-01  1.49174614e-01 -2.46670641e-02  1.53902113e+00
   8.07513104e-02  0.00000000e+00  1.43538993e+01  1.14766974e+01
   5.44707172e-03 -4.02245465e-02  5.49879918e-03  2.86548552e-03
  -1.32545604e+01  2.80153095e+01  1.97007085e+01  0.00000000e+00
   0.00000000e+00  5.17826410e+01 -5.86671543e+00  7.68531598e-03
  -2.36637545e+01 -3.28346221e+00  0.00000000e+00  2.34297052e+01
   8.75243684e+01]]
```

Left ones are ridge regression weights and right ones are for lasso regression. as it can be seen weights for lasso are slightly sparser (sheer zero for some weights) and ridge regression will have sparser weights as the λ increases. So, ridge regression has tried to reach to a performance similar to lasso regression simply with a bigger λ.