3.1 Basic Implementation

Relevant files: *Sec1.py, inpu.py, linear_fit.py*

(Initial weight vector has been set as zero vector)

Output of file: Output\Out1.txt

For *maxit* stopping critera:

α	Final MSE (train)	Final MAE (train)	Final MSE (validation)	Final MAE (validation)
0.1	Diverges	Diverges	Diverges	Diverges
0.01	Diverges	Diverges	Diverges	Diverges
0.001	0.22282572493432173	0.3760290120346339	0.09405022775563182	0.09091900686248729

For *reltol* stopping criteria:

α	Final MSE (train)	Final MAE (train)	Final MSE (validation)	Final MAE (validation)
0.1	Diverges	Diverges	Diverges	Diverges
0.01	Diverges	Diverges	Diverges	Diverges
0.001	0.38881067790449847	0.5011810207678291	0.08217559439206809	0.0820887944933938

If alpha is not sufficiently small, gradient descent algorithm is not able to converge, i.e., MSE becomes arbitrarily large and no minima for cost function is found.

3.2 Ridge Regression

Relevant files: *Sec1.py, inpu.py, linear_fit.py*

Output of file: Output\Out2.txt

Cost function:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N}(y_n - w^T x_n)^2 + \lambda \sum_{j=1}^{2049} w_j^2$$

$$J(w) = \left|\left|Xw - y\right|\right|^2 + \lambda\left|\left|w\right|\right|^2$$

Parameter update function:

$$\nabla_{\mathrm{w}} E_{in}(\mathrm{w}) = \frac{2}{N}\left(\sum_{n=1}^{N}(w^T x_n - y_n)\cdot x_n + \lambda \mathrm{w}\right)$$

α = 0.001

Stopping criteria = *maxit*

it = 1000

Name	Final MSE (train)	Final MAE (train)	Final MSE (validation)	Final MAE (validation)
Linear	0.22282572493432173	0.3760290120346339	0.09405022775563182	0.09091900686248729
Ridge (λ= 5)	0.14687606584412294	0.3002182441295067	0.10733115873786968	0.09781390398739045
Ridge (λ= 25)	0.16633349593063995	0.3214459237457869	0. 10302342975223235	0. 09538660399365138

$\lambda = 25$ gives lesser MSE and MAE than $\lambda = 5$ on validation set, which implies that for the given implementation, $\lambda = 5$ has high bias and $\lambda = 25$ has low bias and low variance.

3.3 Using Scikit-Learn Library

Relevant files: *Sec3.py, inpu.py*

Output of file: Output\Out3.txt

Comparison of sections 3.1 to 3.3:

	Name	Final MSE (train)	Final MAE (train)	Final MSE (validation)	Final MAE (validation)
3.1	Basic Linear	0.22282572493432173	0.3760290120346339	0.09405022775563182	0.09091900686248729
3.2	Ridge (λ= 5)	0.14687606584412294	0.3002182441295067	0.10733115873786968	0.09781390398739045
3.3	Using Scikit	7.285247240934699	2.6991197159323446	1.474345543923734	0.44223740561431435

3.4 Feature Selection

Relevant files: *Sec4.py, inpu.py, linear_fit.py*

Output of file: Output\Out4.txt

Linear regression

α = 0.001

Stopping criteria = *reltol*

bound = 0.000001

#Features	Final MSE (train)	Final MAE (train)	Final MSE (validation)	Final MAE (validation)
2490	0.38881067790449847	0.5011810207678291	0.08217559439206809	0.0820887944933938
10 (SelectKBest)	1.2043872712872432	0.8475841489401378	0.2796538173445502	0.1590609557730694
10 (SelectFromModel)	1.1962115989679691	0.8315110334012059	0.3886584911509137	0.17920046770939935

3.5 Classification

Relevant files: *Sec5.py, inpu.py*

Output of file: Output\Out5.txt

Update function:

For simplicity,

$$a_r = e^{(\theta_r)^T x}$$

$$\frac{da_r}{d\theta_r} = a_r x$$

$$\frac{da_r}{d\theta_p} = 0 \text{ for } (p \neq r)$$

$$\sigma = \sum_{p=1}^{8} a_r$$

$$J(\theta_r) = -y log\left(h_{\theta_r}(x)\right) - (1-y) log\left(1 - h_{\theta_r}(x)\right)$$

$$J(\theta_r) = -ylog(a_r) - (1-y)log(1+\sigma-a_r) + log(1+\sigma)$$

$$\frac{dJ(\theta_r)}{d\theta_r} = -y\frac{1}{a_r}a_r x - 0 + \frac{a_r}{1+\sigma}x$$

$$\frac{dJ(\theta_r)}{d\theta_r} = -yx + h_{\theta_r}(x)x$$

$$\frac{dJ(\theta_r)}{d\theta_r} = (h_{\theta_r}(x) - y)x$$

$$\text{Update Function for } (\theta = \theta_r) = \sum_{n=1}^{N}(h_{\theta_r}(x_n) - y_n)x_n$$

Which is the same update function as linear regression

3.6 Visualisation

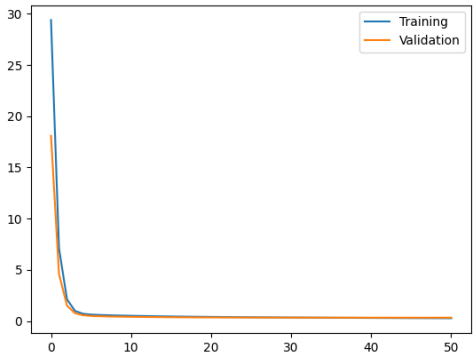
a) Plot training and validation MSE loss
b) Normalization
c) Diving data into different sized groups

it = 50

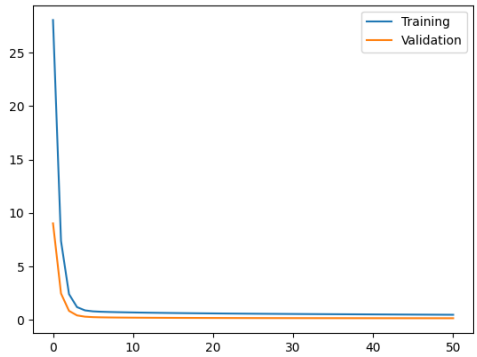
Fraction	Final MSE (Train)*	Final MSE (Validation)*
0.25	0.398625380403785	0.30952612008992164
0.5	0.45743676633836533	0.18074135026665436
0.75	0.5348241543864707	0.11875836442482343
1	0.5520432975275971	0.08741084921216116

With reduced training set, in-sample errors are less since curve is easily able to fit, however out-sample errors are larger.

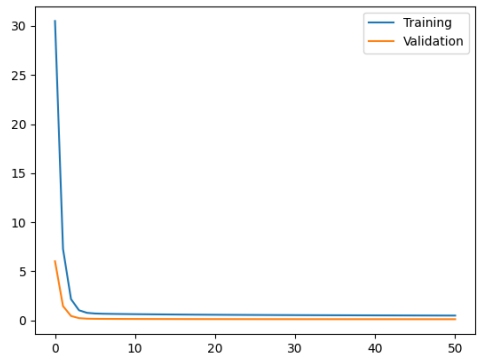
* - Makes use of numpy.random, different values at each run



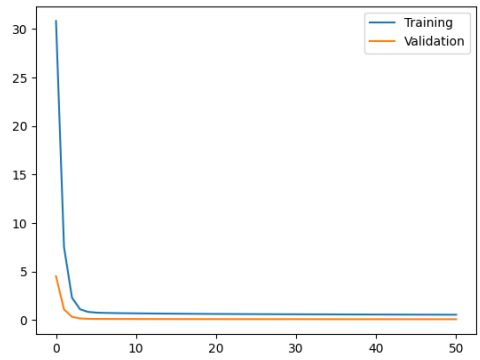
Fraction = 0.25



Fraction = 0.5



Fraction = 0.75



Fraction = 1

d) Dividing into two equal parts

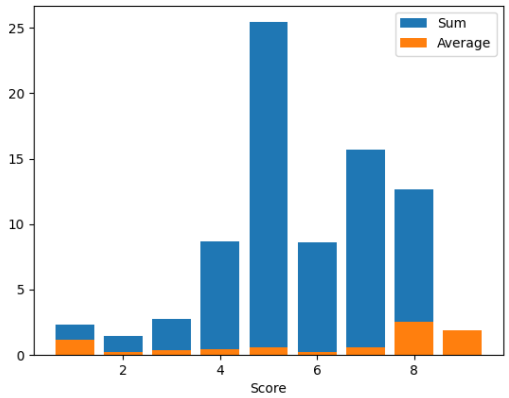
1. For 3.1

Mean absolute difference = 1.457065325480676

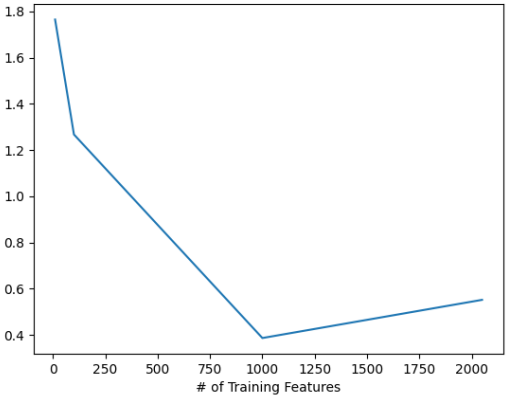
2. For 3.2

Mean absolute difference = 1.4632523576770717

e) Plotting against score values



f) More feature selection



3.7 Generalization Analysis

Relevant files: *Sec7.py, linear_fit.py*

Output of file: Output\Out7.txt

Dimension	$E_{out} - E_{in}$
2	0.5457668016226359
5	0.45152483289549417
10	0.32685555703491354
100	1.4409368603683705

This shows us that generalization bound is a loose estimate for E_{out}.

4 Evaluation

- Relevant files: *main.py*
- Dependencies: *linear_fit.py*