Classification and Regression

CSE574 Introduction to Machine Learning
Spring 2019

Programming Assignment 1

Group No : 33
Manasi Kulkarni (Student No : 50288702)
Priyanka Pai (Student No : 50295903)
Kundan Kumar (Student No : 50301604)

Abstract :

This project involves getting familiar with various types of regression techniques such as linear regression, ridge regression and non-linear regression. The goal of this project is to calculate and minimize errors using each of the techniques mentioned and compare the results.

1. Introduction

Classification involves predicting a label, whereas regression involves predicting a value. Both, classification and regression problems are part of Supervised Machine Learning. The goal is to determine the mapping function : Y = f(X), where Y is the output variable and X is the input variable, due to which we can determine efficient mapping for any set of new input data X.

1.1 Concepts and definitions

Linear Regression

Linear regression is a type of prediction model where the relationships are predicted using linear functions. Such models can be fitted more accurately using least squares method.

Ridge Regression

In ridge regression modeling, we introduce a hyperparameter λ, depending on which our model can be made more fit. We recursively need to calculate the errors using different values of λ and determine at which value, our model will be the best fit. Such a value will be considered optimal.

Non-linear Regression

Non-linear regression is a type of prediction model, where the relationships are predicted using non-linear functions; unlike linear regression. This technique is more useful in case of complex models.

Gradient Descent

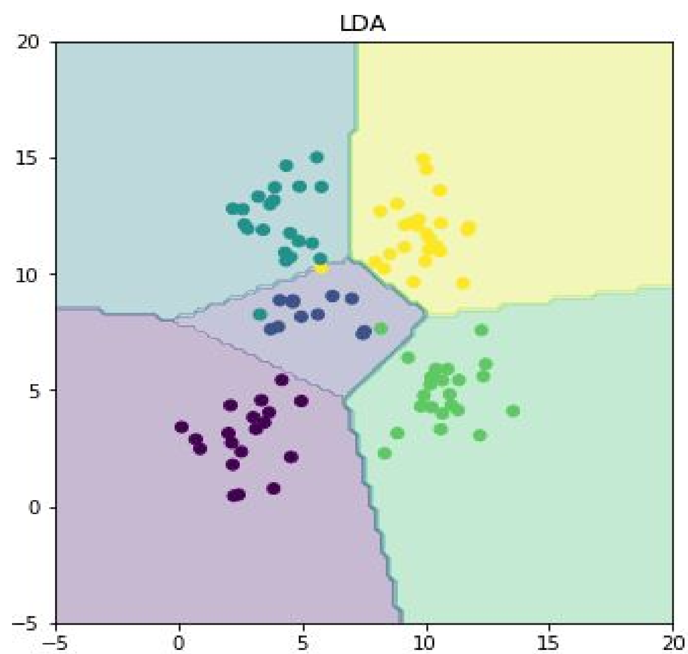
Gradient descent is a type of optimization algorithm, where we find the minimum value of the function. Here, we need to find the minimum error function, so that the model will be a best fit.

2. Implementation

2.1 Problem 1 : Experiment with Gaussian Discriminators

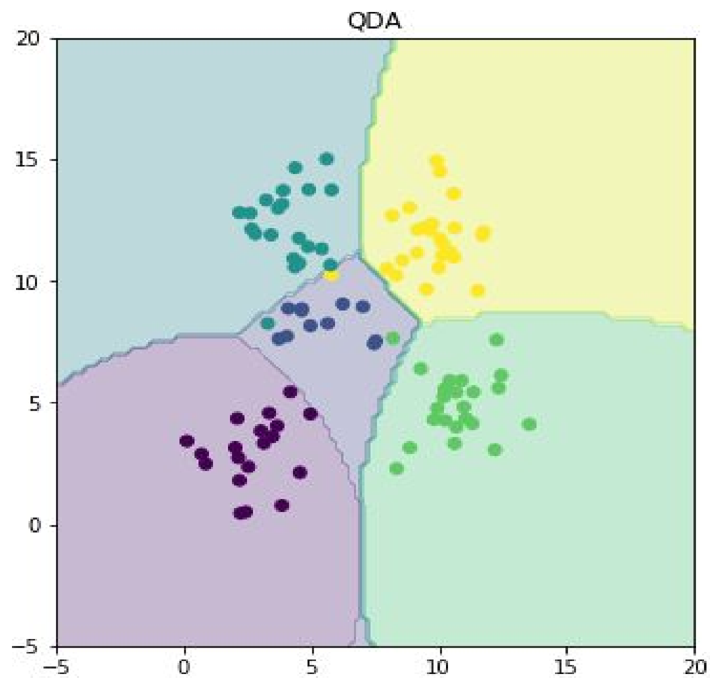
Linear Discriminant Analysis

The boundaries in LDA are straight lines, as there is no quadratic term involved in the computation.
Accuracy : 97%



Quadratic Discriminant Analysis

The boundaries in QDA are curved, as there is quadratic term involved in the computation. This quadratic term comes into picture as we calculate covariance of each class of training data, unlike LDA.
Accuracy : 97%



Observations

1. As the boundaries are curved in QDA, we can use this method to classify **large** and **complex** datasets. Thus, to get more **accurate and better** results, QDA should be preferred. LDA
2. LDA model is less likely to **overfit** than QDA, as it does not account to the complexity and is preferable for small datasets

2.2 Problem 2 : Experiment with Linear Regression

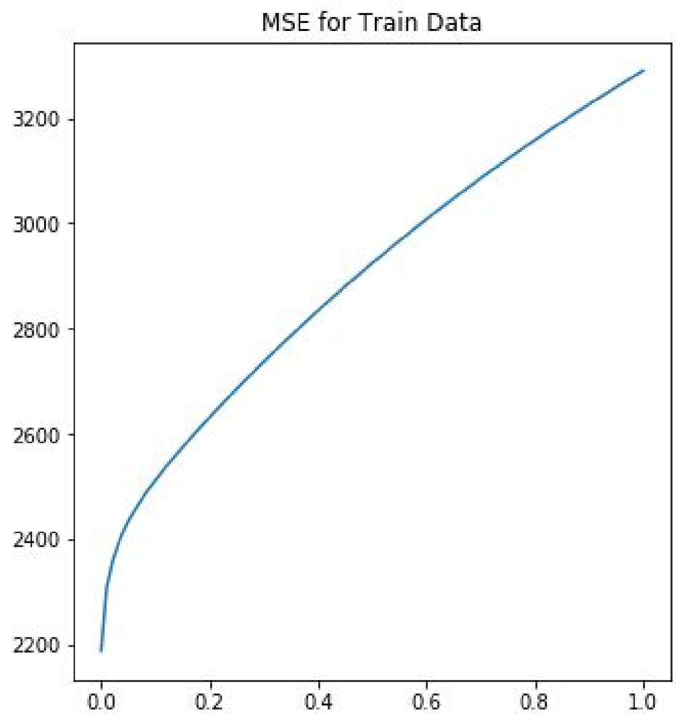
	MSE Without Intercept	MSE With Intercept
Training Data	19099.446844570746	2187.160294930391
Test Data	106775.36155789059	3707.8401813150163

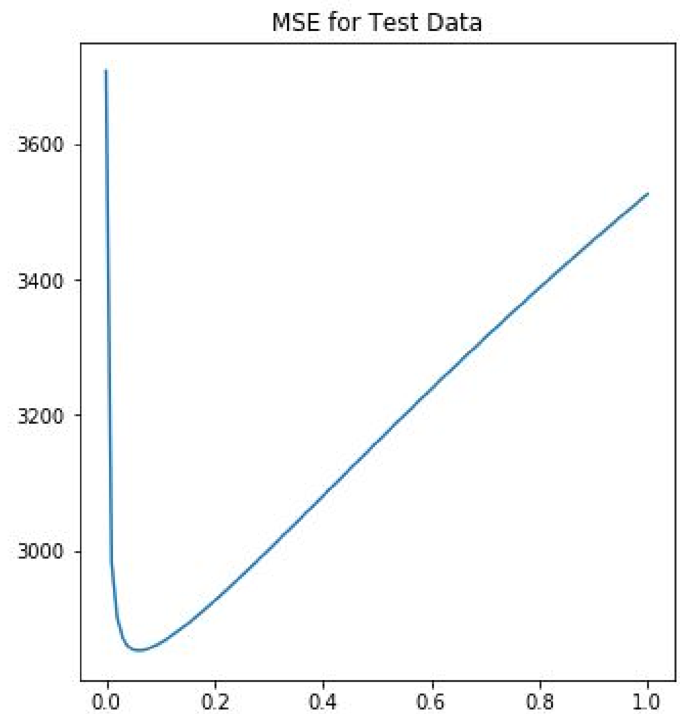
Adding the bias term reduces the error by almost 70%.

Observation

MSE with intercept is better as the error is minimized to a great extent by adding the **bias term**

2.3 Problem 3 : Experiment with Ridge Regression





The above graphs show that **MSE for train data is higher than MSE for test data**

The **values obtained for MSE using Ridge Regression** are as follows :

λ	**Training data**	**Test data**
0	2187.16029493	3707.84018132
0.01	2306.83221793	2982.44611971
0.02	2354.07134393	2900.97358708
0.03	2386.7801631	2870.94158888
0.04	2412.119043	2858.00040957
0.05	2433.1744367	2852.66573517
0.06	**2451.52849064**	**2851.33021344**
0.07	2468.07755253	2852.34999406

0.08	2483.36564653	2854.87973918
0.09	2497.74025857	2858.44442115
0.1	2511.43228199	2862.75794143
0.11	2524.60003852	2867.63790917
0.12	2537.35489985	2872.96228271
0.13	2549.77688678	2878.64586939
0.14	2561.92452773	2884.62691417
0.15	2573.84128774	2890.85910969
0.16	2585.55987497	2897.30665895
0.17	2597.10519217	2903.94112629
0.18	2608.49640025	2910.73937213
0.19	2619.74838623	2917.68216413
0.2	2630.8728232	2924.75322165
0.21	2641.87894616	2931.93854417
0.22	2652.77412633	2939.22592987
0.23	2663.56430077	2946.60462378
0.24	2674.25429667	2954.06505602
0.25	2684.84807809	2961.59864341
0.26	2695.34893502	2969.19763677
0.27	2705.75962912	2976.85500119
0.28	2716.0825067	2984.56432079
0.29	2726.31958674	2992.31972181
0.3	2736.4726296	3000.11580946
0.31	2746.54319109	3007.94761559
0.32	2756.53266482	3015.81055453

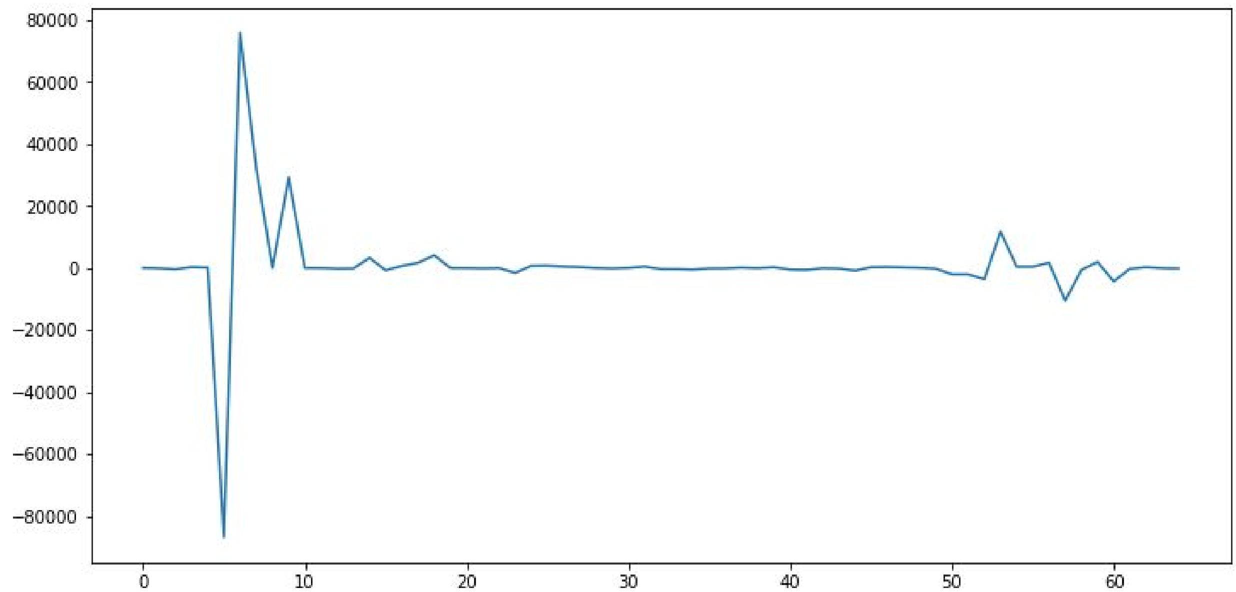
0.33	2766.44231574	3023.70038563
0.34	2776.27330654	3031.61318093
0.35	2786.02671854	3039.54529713
0.36	2795.70356824	3047.49335111
0.37	2805.30482034	3055.45419817
0.38	2814.83139806	3063.42491285
0.39	2824.28419133	3071.40277169
0.4	2833.66406312	3079.38523776
0.41	2842.97185452	3087.36994673
0.42	2852.2083886	3095.35469418
0.43	2861.3744735	3103.33742413
0.44	2870.47090474	3111.31621849
0.45	2879.49846701	3119.28928746
0.46	2888.45793552	3127.25496075
0.47	2897.35007697	3135.21167941
0.48	2906.17565032	3143.15798839
0.49	2914.93540723	3151.09252966
0.5	2923.63009243	3159.01403582
0.51	2932.26044392	3166.92132421
0.52	2940.82719309	3174.81329145
0.53	2949.33106473	3182.68890838
0.54	2957.77277699	3190.54721533
0.55	2966.15304137	3198.38731777
0.56	2974.47256259	3206.20838225
0.57	2982.73203851	3214.00963255

0.58	2990.93215999	3221.79034621
0.59	2999.07361078	3229.5498512
0.6	3007.15706742	3237.28752288
0.61	3015.1831991	3245.00278108
0.62	3023.15266757	3252.69508746
0.63	3031.06612707	3260.36394297
0.64	3038.92422416	3268.00888553
0.65	3046.72759776	3275.6294878
0.66	3054.47687898	3283.22535516
0.67	3062.17269114	3290.79612376
0.68	3069.81564971	3298.34145873
0.69	3077.40636224	3305.86105245
0.7	3084.94542842	3313.354623
0.71	3092.43344001	3320.82191265
0.72	3099.87098085	3328.26268646
0.73	3107.25862691	3335.67673095
0.74	3114.59694628	3343.06385289
0.75	3121.88649919	3350.42387813
0.76	3129.12783807	3357.75665047
0.77	3136.3215076	3365.0620307
0.78	3143.46804472	3372.33989556
0.79	3150.56797875	3379.59013686
0.8	3157.62183137	3386.81266063
0.81	3164.63011677	3394.00738631
0.82	3171.59334168	3401.17424594

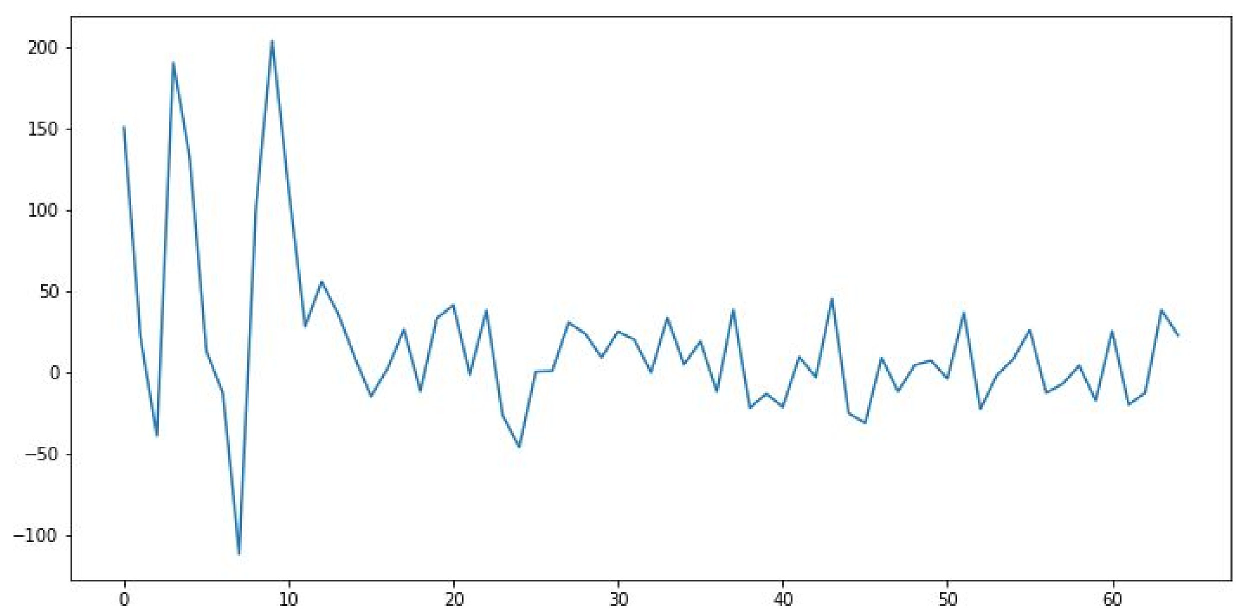
0.83	3178.51200544	3408.31318353
0.84	3185.38660008	3415.42415428
0.85	3192.21761044	3422.50712403
0.86	3199.0055142	3429.56206859
0.87	3205.75078202	3436.58897321
0.88	3212.45387757	3443.58783202
0.89	3219.11525768	3450.55864755
0.9	3225.73537241	3457.50143021
0.91	3232.31466512	3464.41619786
0.92	3238.8535726	3471.30297539
0.93	3245.35252514	3478.16179431
0.94	3251.81194665	3484.99269234
0.95	3258.23225474	3491.79571308
0.96	3264.61386081	3498.57090566
0.97	3270.95717015	3505.3183244
0.98	3277.26258207	3512.03802854
0.99	3283.53048993	3518.7300819
1	3289.7612813	3525.39455263

Thus, it can be observed from the table that when $\lambda = 0.06$, the MSE is minimum for test data. Thus, **$\lambda = 0.06$** is the optimal value of lambda.

Comparison of relative magnitudes of weights learnt using OLE regression(from Problem 2) and Ridge Regression :



Weights learnt using OLE regression with intercept



Weights learnt using Ridge regression with intercept

From the above 2 plots, we can observe that OLE regression has larger weights as compared to ridge regression. It can be concluded that ridge regression can be used for faster computations.

Observations

1. MSE on test data is much lower after using ridge regression, as compared to linear regression.
2. Weights obtained using linear regression are quite **higher in magnitude** as compared to ridge regression.

2.4 Problem 4 : Using Gradient Descent for Ridge Regression Learning

The values obtained for **MSE using Gradient Descent** are as follows :

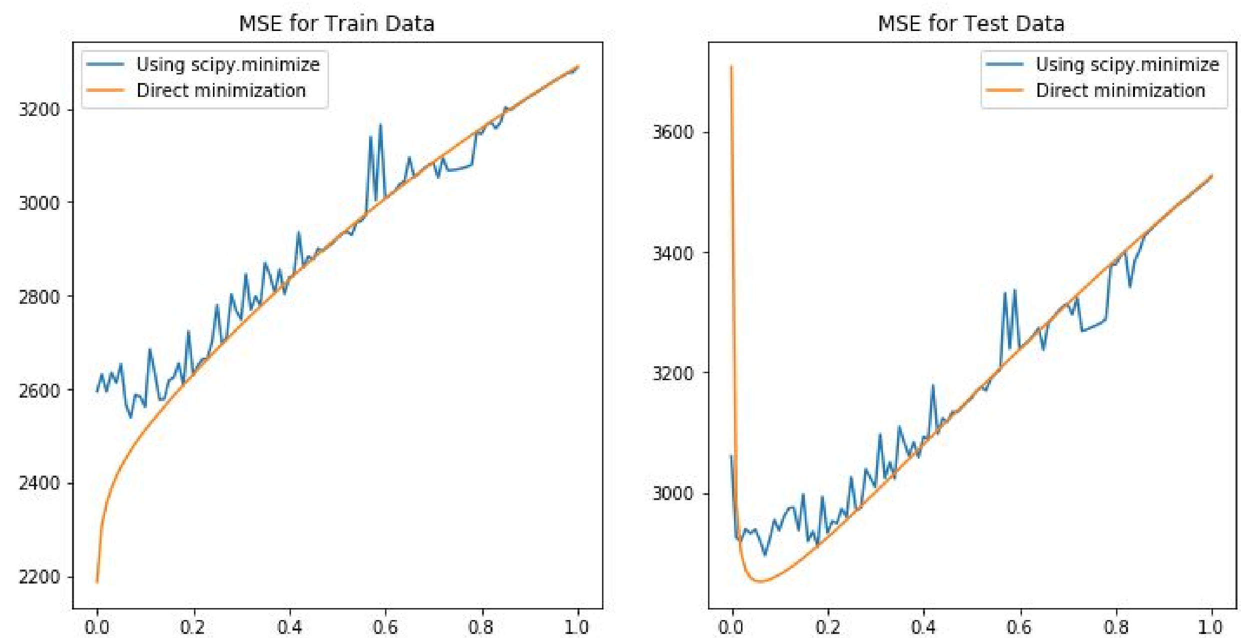
λ	**Training data**	**Test data**
0	2338.91992449	2937.08989232
0.01	2312.99264189	3004.73257482
0.02	2366.05089479	2884.46447618
0.03	2395.82663851	2861.0768814
0.04	2419.22214468	2861.1419262
0.05	2428.29961836	2853.65188958
0.06	**2448.72576438**	**2851.16222054**
0.07	2468.50464221	2835.74844394
0.08	2488.5319061	2864.25110163
0.09	2498.4346125	2857.2354291
0.1	2524.72737978	2860.42954807
0.11	2545.74193734	2892.59734183
0.12	2547.80826403	2883.91423062
0.13	2538.63933612	2872.69113794
0.14	2576.80112385	2905.27874561
0.15	2578.64718461	2914.61681039
0.16	2585.17923633	2896.96281327

0.17	2601.19614639	2908.00100069
0.18	2607.66225277	2909.39342805
0.19	2613.93543998	2920.13581249
0.2	2627.67104083	2928.63297145
0.21	2627.69417219	2944.90202925
0.22	2663.71060218	2948.34449844
0.23	2664.53805622	2965.51658289
0.24	2682.70908158	2962.79912695
0.25	2687.18343329	2968.39106166
0.26	2695.20000046	2970.06166079
0.27	2710.01944902	2974.49925009
0.28	2715.64502949	2983.84742813
0.29	2723.39433845	2991.25320421
0.3	2735.97555856	3003.7420779
0.31	2745.0594986	3009.47654181
0.32	2755.70932604	3016.88025862
0.33	2766.381322	3014.1230042
0.34	2777.92531452	3022.79626412
0.35	2782.8635533	3036.71477686
0.36	2793.43675759	3042.63639832
0.37	2805.44085612	3060.73965132
0.38	2791.91929861	3050.41931929
0.39	2803.50030849	3058.38771358
0.4	2838.42847041	3092.06227882
0.41	2842.70700486	3089.12329626

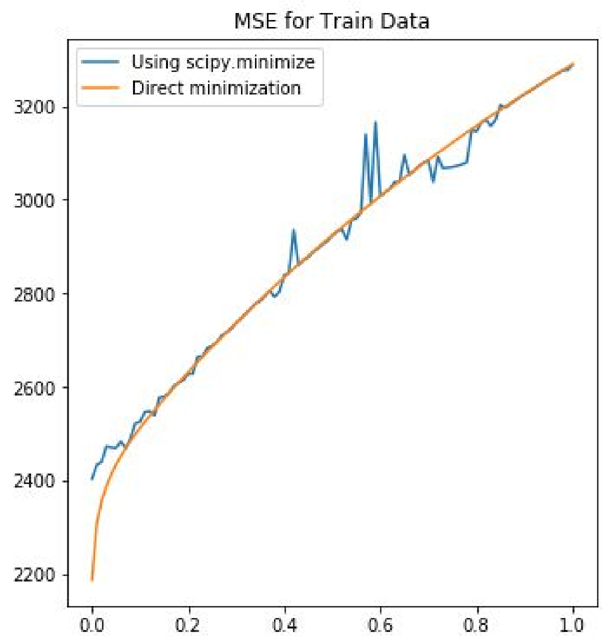
0.42	2934.98459478	3178.0782617
0.43	2859.47802993	3097.09536166
0.44	2870.25264005	3110.90474595
0.45	2876.639375	3116.78667708
0.46	2888.10587316	3126.62321591
0.47	2895.01551142	3133.06420287
0.48	2903.8550424	3141.25787724
0.49	2911.17830908	3149.65797446
0.5	2922.55547376	3155.95194324
0.51	2931.39809118	3167.66716733
0.52	2936.64908709	3174.9423542
0.53	2914.13915007	3153.3007165
0.54	2956.55343006	3188.37208987
0.55	2959.29935628	3197.56504434
0.56	2972.0602193	3203.8884186
0.57	3139.36666004	3331.31281653
0.58	2990.22887819	3221.51735567
0.59	3164.84849761	3336.34385524
0.6	3007.0849066	3237.16505687
0.61	3015.22833946	3245.63762946
0.62	3023.13651661	3252.36583201
0.63	3038.00523558	3262.01965913
0.64	3038.22903389	3267.69738105
0.65	3095.25086829	3236.91272399
0.66	3053.06513585	3281.44028545

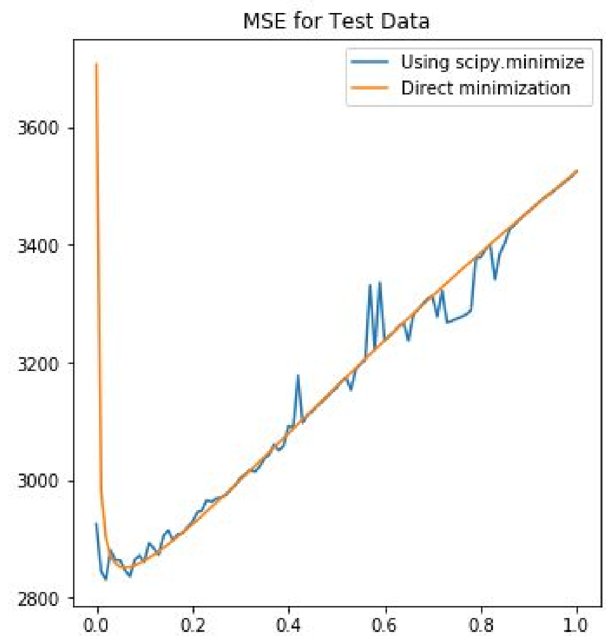
0.67	3061.06512317	3290.33014229
0.68	3071.27476447	3300.70854684
0.69	3078.60916531	3309.03779347
0.7	3083.75813359	3313.45743109
0.71	3037.84542242	3277.74978865
0.72	3091.81349711	3322.26796225
0.73	3067.6212923	3268.06882427
0.74	3068.44180014	3270.92323817
0.75	3069.77325931	3274.43007392
0.76	3072.17147962	3277.49814648
0.77	3075.23306631	3281.67784405
0.78	3079.59763775	3288.22902798
0.79	3149.39280751	3377.7105921
0.8	3145.74236439	3379.088009
0.81	3163.93801197	3392.17156037
0.82	3171.6514668	3401.25494157
0.83	3156.94815468	3341.56534602
0.84	3170.6121256	3385.2112934
0.85	3202.41234721	3402.25032505
0.86	3196.62585107	3426.10590293
0.87	3203.76477571	3433.86801472
0.88	3211.0478935	3442.98668373
0.89	3218.61277101	3449.90017793
0.9	3225.13482992	3456.86035304
0.91	3231.0089491	3464.13986997

0.92	3237.84066259	3471.74371997
0.93	3244.46977035	3478.7872322
0.94	3251.15038604	3484.45318142
0.95	3257.55258676	3489.91044532
0.96	3263.89579202	3497.14117432
0.97	3270.1920954	3503.79310517
0.98	3276.44729655	3510.41736075
0.99	3277.47617685	3516.94542867
1	3288.88323287	3526.01688464



Plots when ‘maxiter’ are set to 20(preferred value)





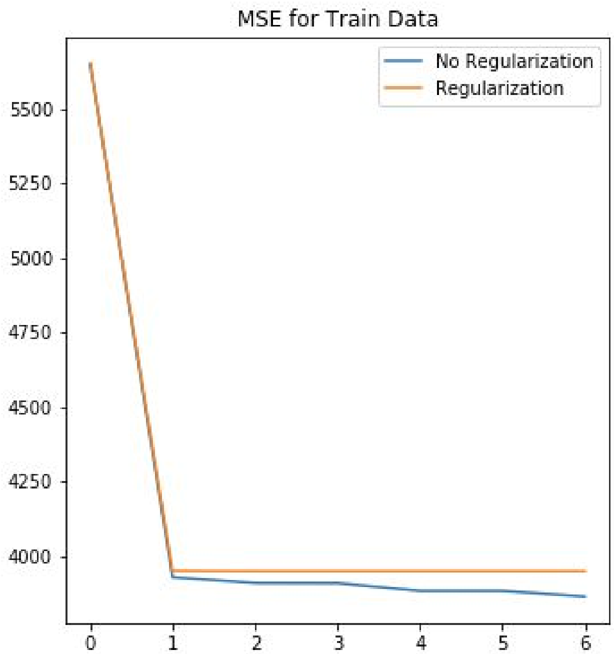
Plots when 'maxiter' are set to 50

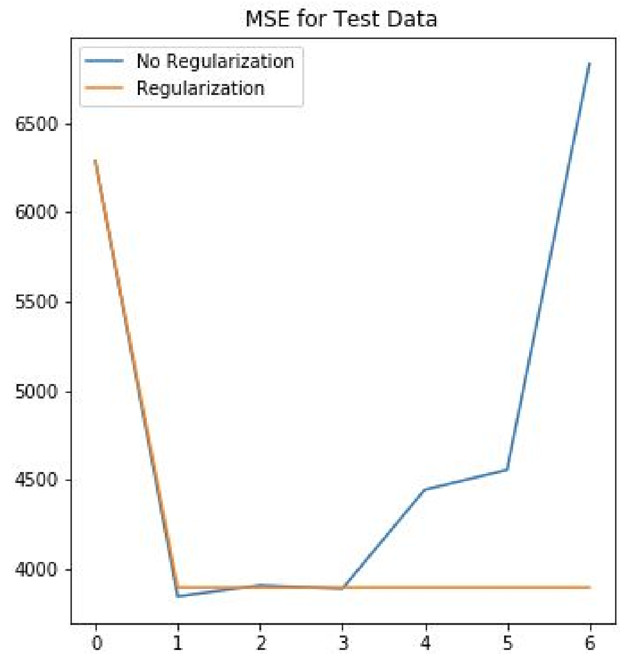
As it can be inferred from the above graphs, using gradient descent for ridge regression follows the same plot as without using gradient descent. The only difference is that the line generated using gradient descent is **not completely smooth** for few values of lambda; for rest of the values of lambda it is smooth.
As we increase the number of iterations, the line gets smoother. For 50 iterations, the spikes are lesser than 20 iterations.

Observation

The curve plotted using gradient descent gets **smoother** as we increase the number of iterations.

2.5 Problem 5 : Non-linear Regression





1. The graph on the left shows MSE using non-linear regression on training data and the graph on the right shows the same on test data.
2. When $\lambda = 0$(no regularization), the error is constant until p=3 and it increases substantially as p increases.
3. On the other hand, after applying regularization, the **error stays constant** throughout all the values of p : from p=1 to p=6

Following table shows comparison of training data and test data errors when $\lambda = 0$ and λ is optimal(set to optimal lambda obtained from Problem 3) :

	λ = 0 (no regularization)		λ = optimal	
p	**Training data**	**Test data**	**Training data**	**Test data**
0	5650.7105389	6286.40479168	5650.71190703	6286.88196694
1	3930.91540732	**3845.03473017**	3951.83912356	**3895.85646447**
2	3911.8396712	3907.12809911	3950.68731238	**3895.58405594**
3	3911.18866493	3887.97553824	3950.68253152	**3895.58271592**
4	3885.47306811	4443.32789181	3950.6823368	**3895.58266828**
5	3885.4071574	4554.83037743	3950.68233518	**3895.5826687**
6	**3866.88344945**	6833.45914872	**3950.68233514**	**3895.58266872**

The **optimal value of p** for test data without regularization is **1**.
For test data using **optimal value of lambda**, the mse is almost the same when **p >= 1**

2.6 Problem 6 : Interpreting Results

Approach	Training Data	Test Data
Linear Regression with intercept	**2187.160294930391**	3707.8401813150163
Linear Regression without intercept	19099.446844570746	106775.36155789059
Ridge Regression (for optimal lambda)	2451.52849064	2851.33021344
Ridge Regression using Gradient Descent	2448.72576438	**2851.16222054**
Non-linear Regression (with regularization for optimal value of p)	3950.68233514	3895.58266872

Observations

1. On training data, we get the minimum value for MSE using Linear regression model with intercept. Therefore, linear regression with intercept is the best way to predict data using input features for training data.
2. However, on testing data, we find that using Gradient Descent method for Ridge Regression produces the least MSE and Ridge regression using optimal value of lambda produces almost the same MSE.
3. For predicting diabetes level using input features, we need to consider optimal models on testing data, as the test data is used to assess our model, which is a validation of how well we have trained it.
4. Thus, **Ridge regression using either optimal value of lambda or Gradient Descent** is the best setting that can be used to predict data.