DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY AT BUFFALO

CSE 574 Programming Assignment 2
Classification and Regression

Group 100
Luigi Di Tacchio Zakieh Hashemifar Alexander Simeonov
April 9, 2015

1 PROBLEM 1 - EXPERIMENT WITH GAUSSIAN DISCRIMINATORS

For this problem, in order to calculate a single covariance matrix for all the classes for LDA, we could simply calculate the covariance matrix of the whole data. But we used another approach which seems more reasonable. We used a weighted averaging of the different covariance matrices. We tried both of them with the sample data and the resulting accuracy was the same. But we think that our approach should work better in general.

1.1 REPORT THE ACCURACY OF LDA AND QDA ON TEST DATA

LDA accuracy = 97%
QDA accuracy = 94%

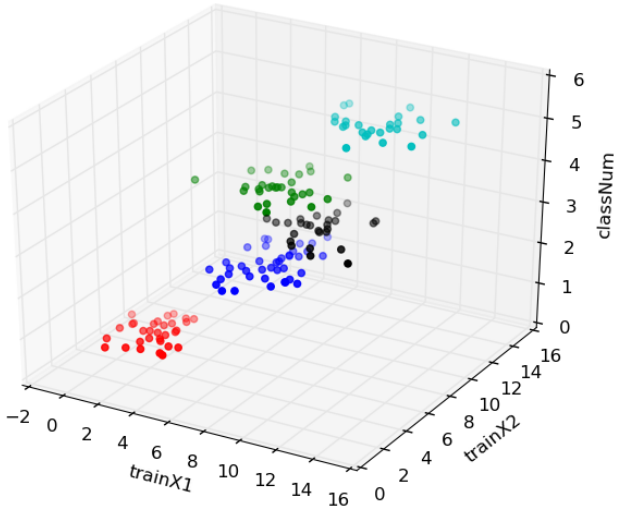

Figure 1.1: Training set

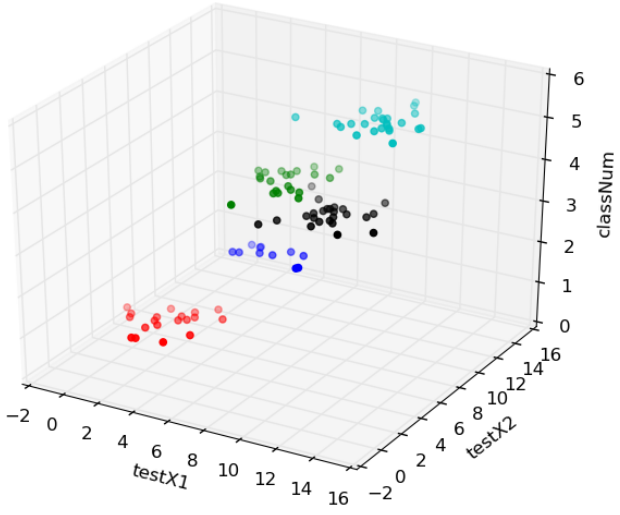

Figure 1.2: Test set

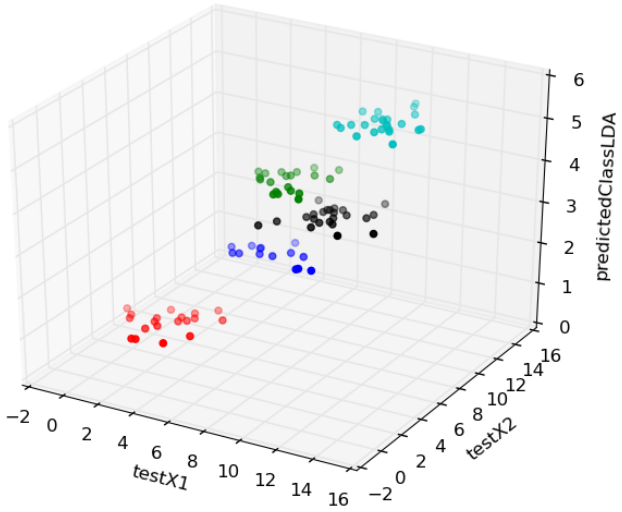

Figure 1.3: LDA and predicted classes

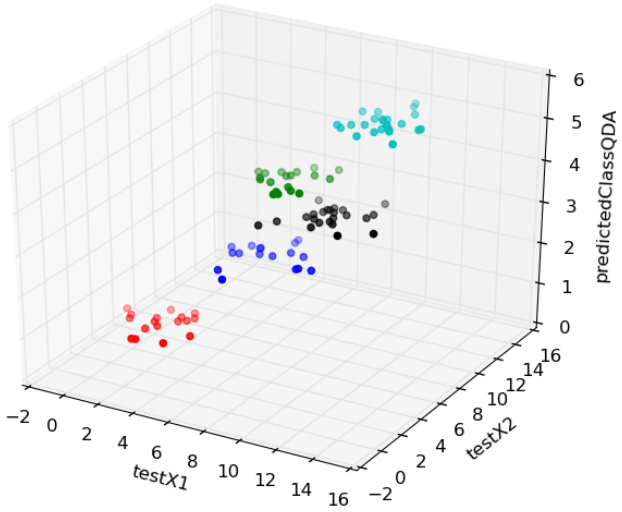

Figure 1.4: QDA and predicted classes

As the scatter diagram of training data shows, class two (blue dots) seems more widely distributed. When using LDA, since the covariance of all the classes is the same, this doesn?t harm much and the accuracy is high. But when using QDA, the covariance of class two gets bigger and class one (red dots) gets smaller, since they seem more concentrated. So this causes some of class one points to get classified as two and lead to accuracy decrement.

1.2 THE DISCRIMINATING BOUNDARY OF LDA AND QDA

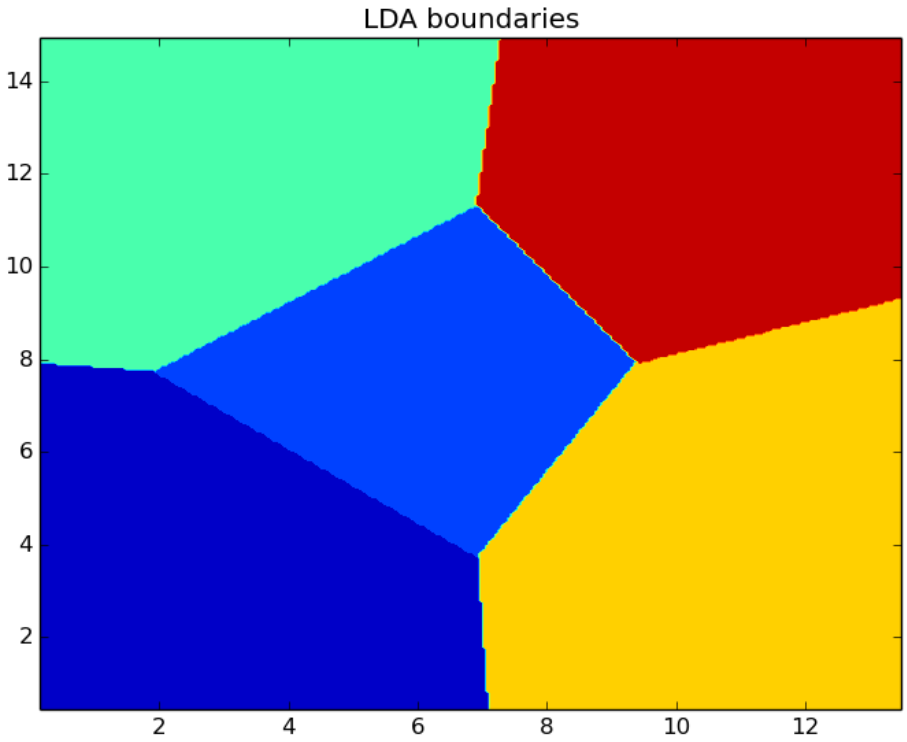

Figure 1.5: LDA boundaries

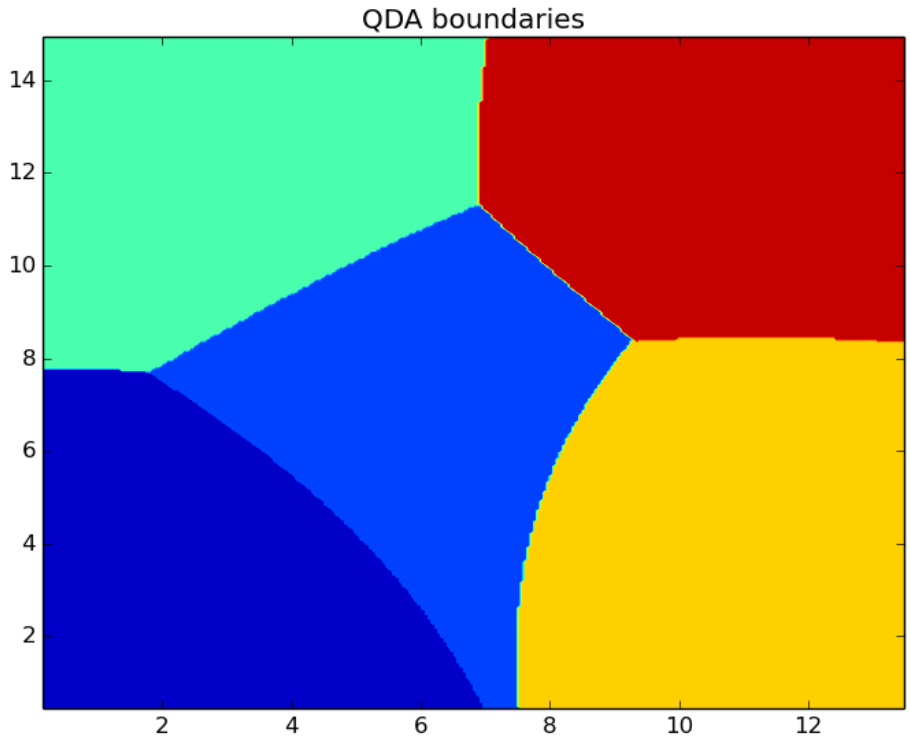

Figure 1.6: QDA boundaries

For LDA, we assign the same covariance matrix to all the classes. So the discriminating boundaries will show up as straight lines. But for QDA, the discriminating boundaries are curved lines, because the covariance matrix of each class differs from the others and the pace of pdf changes is not the same for all of them.

2 Problem 2 - Experiment with Linear Regression

	without intercept	with intercept
Training data	8.88388057487	3.0063021236
Test data	23.1057743271	4.30571723465

Table 2.1: RMSE for training and test data with and without intercept

The RMSE values obtained from the simulations are shown in table 2.1. They highlight two positive effects of using an intercept for prediction: a significant error decrease when considering a single data set in isolation (either training or test), and an even more impressive

reduction of the error committed on the test set compared to the training set.

More extensively, we can see that the RMSE on the training data is reduced by more than 66% when using an intercept, and similarly the error on the test data decreases by more than 81%.

Moreover, when comparing the RMSE for training and test data, we can see that not using an intercept causes the test error to be 2.6 times the error on the training set. However, when using an intercept, the test error is 1.43 times the RMSE on the training set.

3 Problem 3 - Experiment with Ridge Regression

3.1 Calculate and report the RMSE for training and test data using the testOLERegression function that you implemented in problem 2

Lambda	Testing RMSE	Training RMSE		Lambda	Testing RMSE	Training RMSE
0.0	4.30571723468	3.0063021236		0.0022	3.99034867868	3.4922362047
4e-05	3.8646228359	3.08616671082		0.00224	3.99510918789	3.4970595144
8e-05	3.81049726836	3.1172674643		0.00228	3.99985326505	3.50184229069
0.00012	3.79001316879	3.13868757834		0.00232	4.00458050302	3.50658507534
0.00016	3.7809515878	3.15518691782		0.00236	4.00929052735	3.51128840227
0.0002	3.77702915146	3.16881598247		0.0024	4.01398299427	3.51595279759
0.00024	**3.7758233193**	**3.18062699378**		0.00244	4.01865758873	3.52057877965
0.00028	3.77622035095	3.19121702409		0.00248	4.02331402263	3.52516685908
0.00032	3.77764400798	3.20094914788		0.00252	4.02795203314	3.52971753886
0.00036	3.77976909903	3.21005578115		0.00256	4.0325713811	3.53423131435
0.0004	3.78239921317	3.21869174449		0.0026	4.0371718496	3.53870867338
0.00044	3.7854089347	3.22696350298		0.00264	4.04175324257	3.54315009624
0.00048	3.78871430411	3.23494619823		0.00268	4.04631538353	3.54755605581
0.00052	3.79225674488	3.24269402656		0.00272	4.05085811434	3.55192701758
0.00056	3.79599385976	3.25024680348		0.00276	4.05538129411	3.55626343972
0.0006	3.79989393137	3.25763424847		0.0028	4.05988479811	3.56056577317
0.00064	3.80393251537	3.26487885773		0.00284	4.0643685168	3.56483446167
0.00068	3.80809026272	3.2719978751		0.00288	4.06883235491	3.56906994189
0.00072	3.81235148889	3.27900467148		0.00292	4.07327623051	3.57327264347
0.00076	3.8167032102	3.28590972725		0.00296	4.07770007426	3.57744298911
0.0008	3.82113448015	3.29272134258		0.003	4.08210382859	3.58158139465
0.00084	3.82563592274	3.29944615792		0.00304	4.08648744699	3.58568826917
0.00088	3.83019939781	3.30608954004		0.00308	4.09085089333	3.58976401506
0.00092	3.8348177566	3.31265587116		0.00312	4.09519414124	3.59380902814
0.00096	3.83948466002	3.31914876762		0.00316	4.09951717347	3.59782369771
0.001	3.84419444112	3.32557124662		0.0032	4.10381998135	3.60180840665
0.00104	3.8489419994	3.33192585422		0.00324	4.10810256425	3.60576353157
0.00108	3.85372271816	3.33821476427		0.00328	4.1123649291	3.60968944281
0.00112	3.85853239886	3.34443985534		0.00332	4.11660708987	3.61358650464
0.00116	3.86336720828	3.35060277076		0.00336	4.12082906718	3.61745507526
0.0012	3.86822363529	3.35670496578		0.0034	4.12503088785	3.62129550696
0.00124	3.87309845507	3.36274774468		0.00344	4.12921258452	3.6251081462
0.00128	3.87798869905	3.36873229007		0.00348	4.13337419527	3.62889333368
0.00132	3.88289162941	3.3746596862		0.00352	4.13751576329	3.63265140447
0.00136	3.88780471724	3.38053093734		0.00356	4.14163733652	3.63638268811
0.0014	3.89272562354	3.38634698254		0.0036	4.14573896737	3.64008750866
0.00144	3.89765218267	3.39210870732		0.00364	4.1498207124	3.64376618485
0.00148	3.90258238777	3.39781695299		0.00368	4.15388263207	3.64741903013
0.00152	3.90751437777	3.40347252412		0.00372	4.15792479048	3.65104635278
0.00156	3.91244642588	3.40907619455		0.00376	4.16194725508	3.65464845604
0.0016	3.91737692915	3.41462871206		0.0038	4.16595009651	3.65822563811
0.00164	3.92230439915	3.42013080223		0.00384	4.16993338832	3.66177819234
0.00168	3.92722745349	3.42558317147		0.00388	4.1738972068	3.66530640726
0.00172	3.93214480806	3.4309865094		0.00392	4.17784163077	3.66881056668
0.00176	3.93705527005	3.4363414908		0.00396	4.1817667414	3.67229094979
0.0018	3.94195773147	3.44164877712		0.004	4.18567262204	3.67574783124
0.00184	3.94685116329	3.44690901768				
0.00188	3.95173460996	3.45212285065				
0.00192	3.95660718443	3.45729090377				
0.00196	3.96146806356	3.46241379496				
0.002	3.96631648382	3.46749213278				
0.00204	3.97115173737	3.47252651678				
0.00208	3.97597316838	3.47751753779				
0.00212	3.98078016966	3.48246577815				
0.00216	3.98557217948	3.48737181185				

Figure 3.1: Error for different values of lambda

3.2 Plot the error for training and test data for different values of lambda

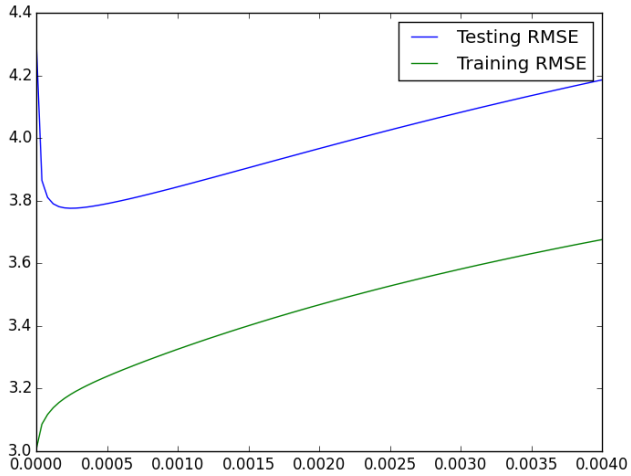

Figure 3.2: Error for different values of lambda

3.3 Compare the relative magnitudes of weights learnt using OLE and weights learnt using ridge regression

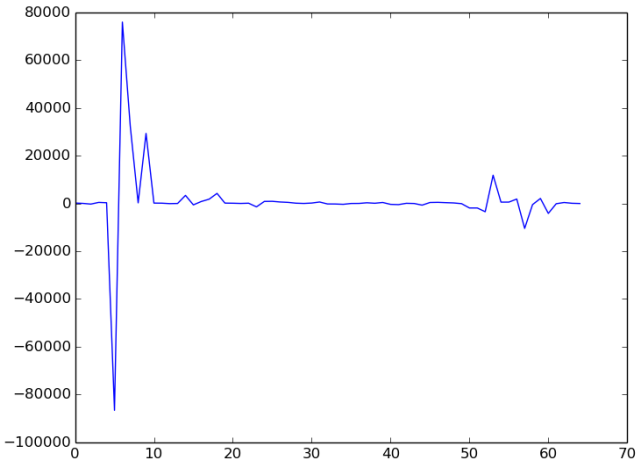

Figure 3.3: OLE weights with intercept

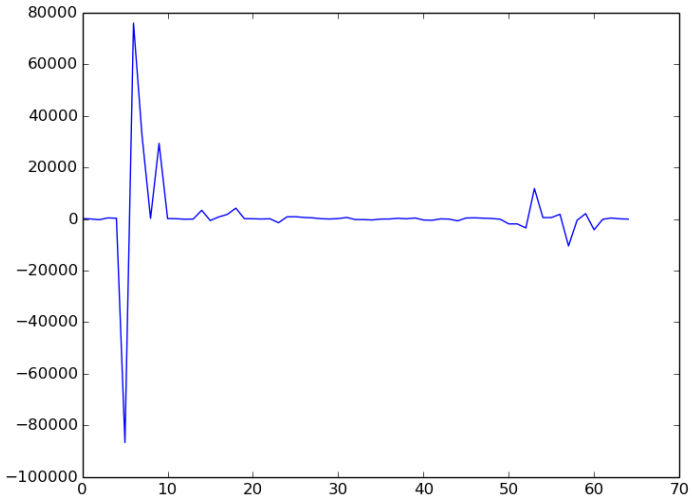

Figure 3.4: Ridge Regression Weights for lambda at 0 with intercept

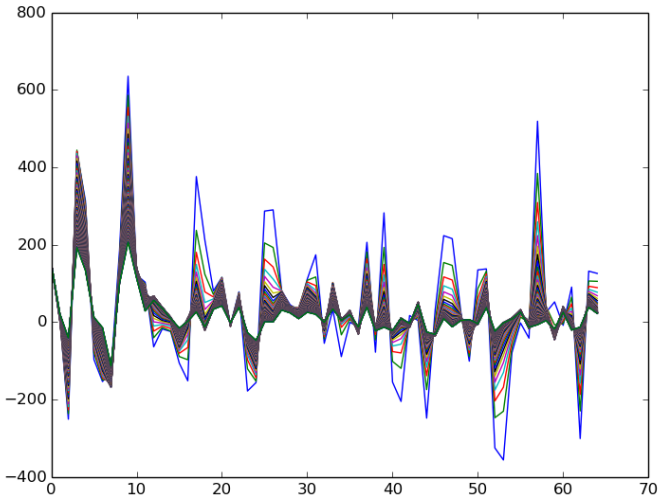

Figure 3.5: Ridge Regression Weights for lambdas between 0.00004 and 0.004 with intercept

3.4 Observations

OLE and Ridge regression produce identical results for their weights when lambda is 0 for Ridge regression which is to be expected. However the magnitude of weights is much tighter when it comes to Ridge regression for lambdas between 0.00004 and 0.004. This is due to the regularization factor that in Ridge regression favors small weights.

3.5 Compare the two approaches in terms of error on training and test data

The RMSE for OLE train data with intercept is: 3.0063021236
The RMSE for OLE test data with intercept is: 4.30571723466
The RMSE for ridge regression (with intercept) can be seen in the table above.

It is clear that the RMSE for Ridge regression is significantly lower for the test data than the one for OLE thus Ridge regression is definitely a better algorithm for the task at hand.

3.6 What is the optimal value of lambda and why?

The value of the optimal lambda is 0.00024 has been additionally highlighted in the RMSE data table shown above. This value is optimal because it minimizes the testing RMSE for our data set.

4 Problem 4 - Using Gradient Descent for Ridge Regression Learning

4.1 Plot the errors on training and test data obtained using the gradient descent learning by varying the regularization parameter lambda

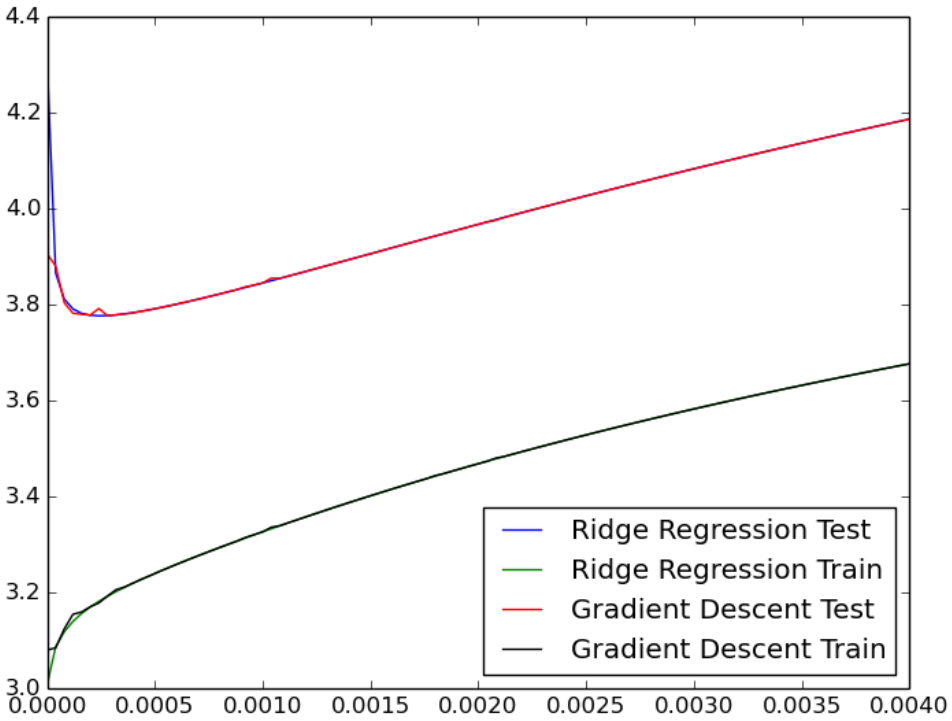

Figure 4.1: RMSE for Ridge Regression and Gradient Descent

4.2 Compare with the results obtained in problem 3

The results obtained in problem 3 are nearly identical to those obtained in problem 4. It should be noted that the lines produced using gradient descent are not as smooth as those produced using regular Ridge regression, having some outliers in a couple of places, though they are few and minor.

One area where problem 3 excels in this data set is that it is much faster than gradient descent, as the minimize function takes a while to converge. However, with bigger matrices, calculating the weights through matrix inversion can be computationally expensive, in some cases even problematic, when the matrix is singular. In such scenarios, Ridge regression

through gradient descent is perhaps a better option since each step is easy to compute.

With this specific data set, direct computation of the weights is faster.

5 Problem 5 - Non-linear Regression

Figure 5.1 shows the RMSE on the training set for different higher order polynomials of the input features. The degree of the polynomials varies from 0 to 6. The prediction error has been computed for two cases: without regularization, and with regularization with optimal value of lambda = 0.00028 (computed in problem 3).

We can see that in both cases the prediction error decreases when increasing the polynomial degree. This is due to the fact that we train using the same data set, therefore higher order curves will fit better the data points and reduce the error. However, this has an impact on the RMSE on the test set, as we will describe in the following paragraph.

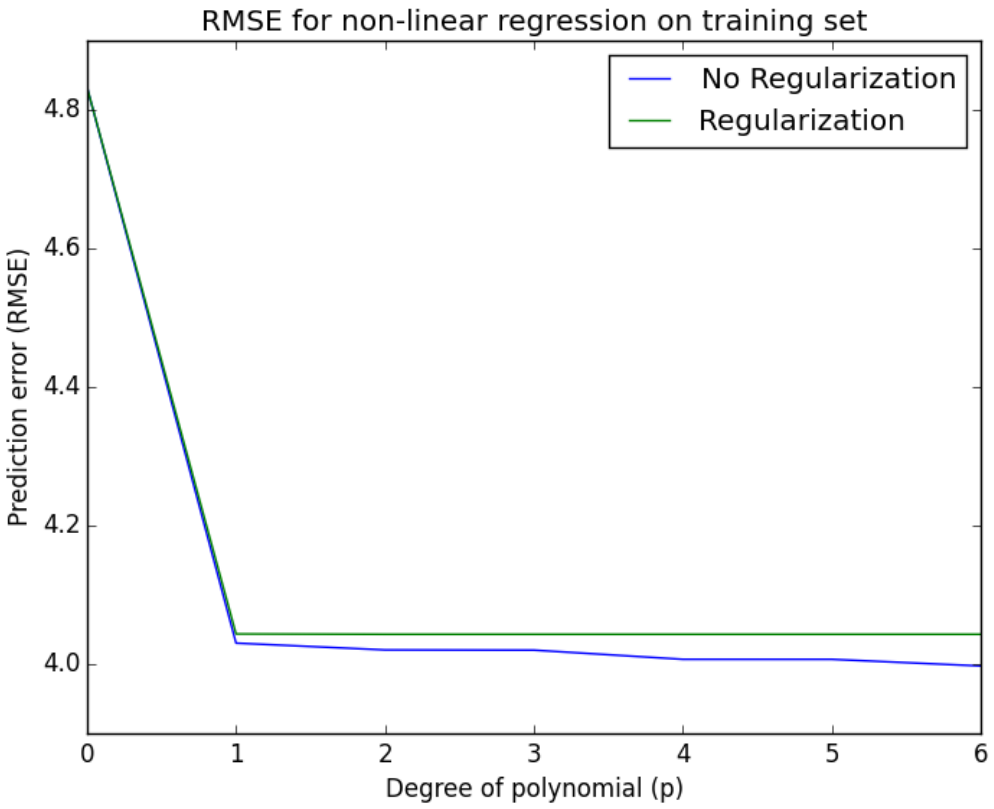

Figure 5.1: RMSE for non linear regression on the training set

Figure 5.2 shows the RMSE on the test set for different higher order polynomials of the input features. As in the previous case, we plotted the error both with and without regularization for different degrees polynomials.

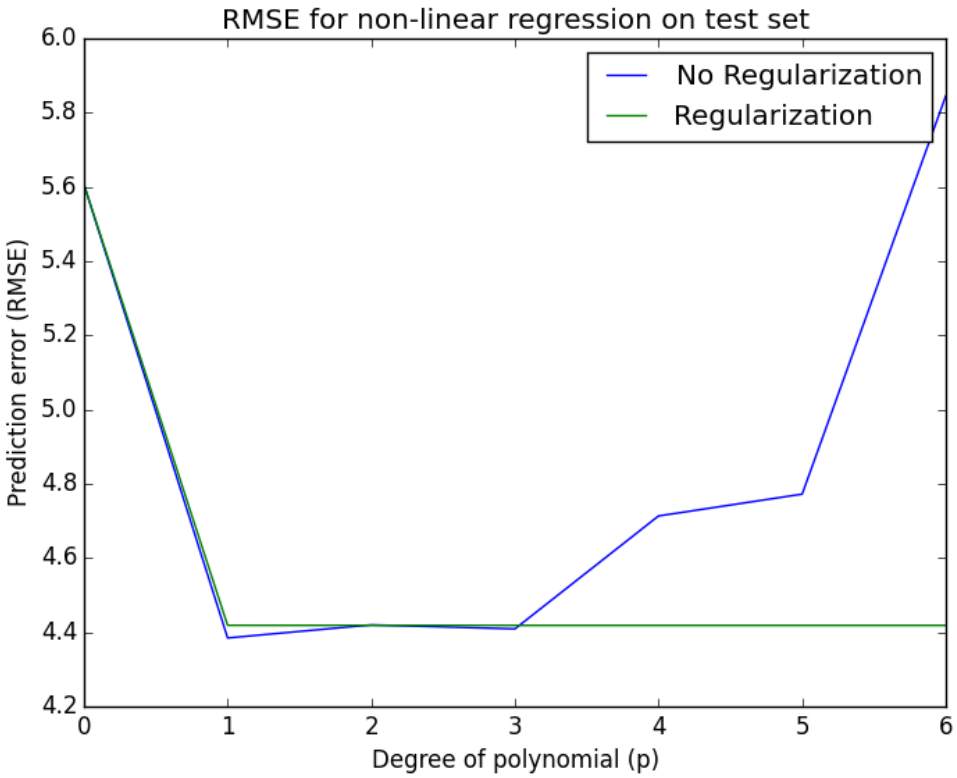

Figure 5.2: RMSE for non linear regression on the test set

When we do not use regularization, the error reaches its minimum at p=1, in other words when we use regular regression. Then it slowly increases until p=3, after which it rapidly grows and becomes even greater than p=0, case in which we use a horizontal line. The reason is that with higher order polynomials we have overfitting: the training error decreases, as figure 5.1 shows, but the learned curve is highly bound to the training data, and with a different data set the error steeply increases.

Using regularization the scenario is different: the error in this case decreases and reaches its minimum at p=5. However, it does not decrease much, especially when compared to the case with no regularization. This is due to the fact that the regularization term penalizes high weights: therefore, even with high order polynomials, the correspondent weights with be very low and the resulting curve very smooth and 'linear'. Simpler curves work better in the general case, with data sets other than the training (Occam's razor).

5.1 BEST P WITHOUT REGULARIZATION

As shown in figure 5.3, while the RMSE on the training set almost imperceptibly decreases,

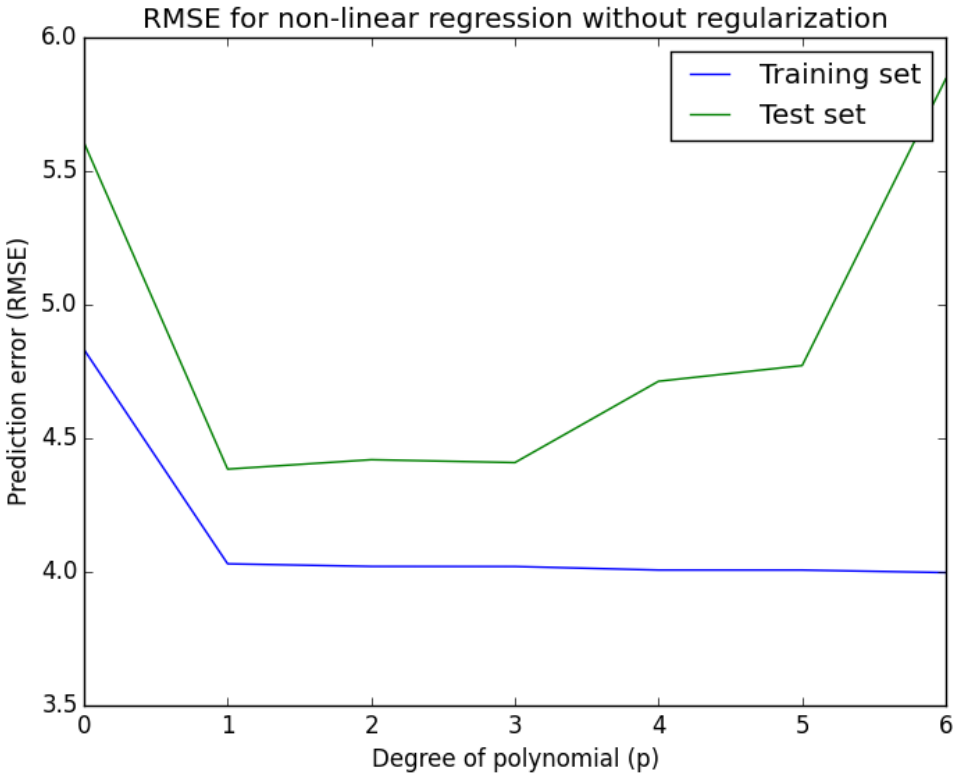

Figure 5.3: RMSE without regularization

reaching a minimum when p=6, the error on the test set visibly increases when using higher order polynomials. Its minimum is at p=1.

The optimal value is therefore at p=1, although 2 and 3 are still acceptable in terms of error on test and training set.

5.2 BEST P WITH REGULARIZATION

Using regularization, the effect of higher order polynomials becomes almost imperceptible.

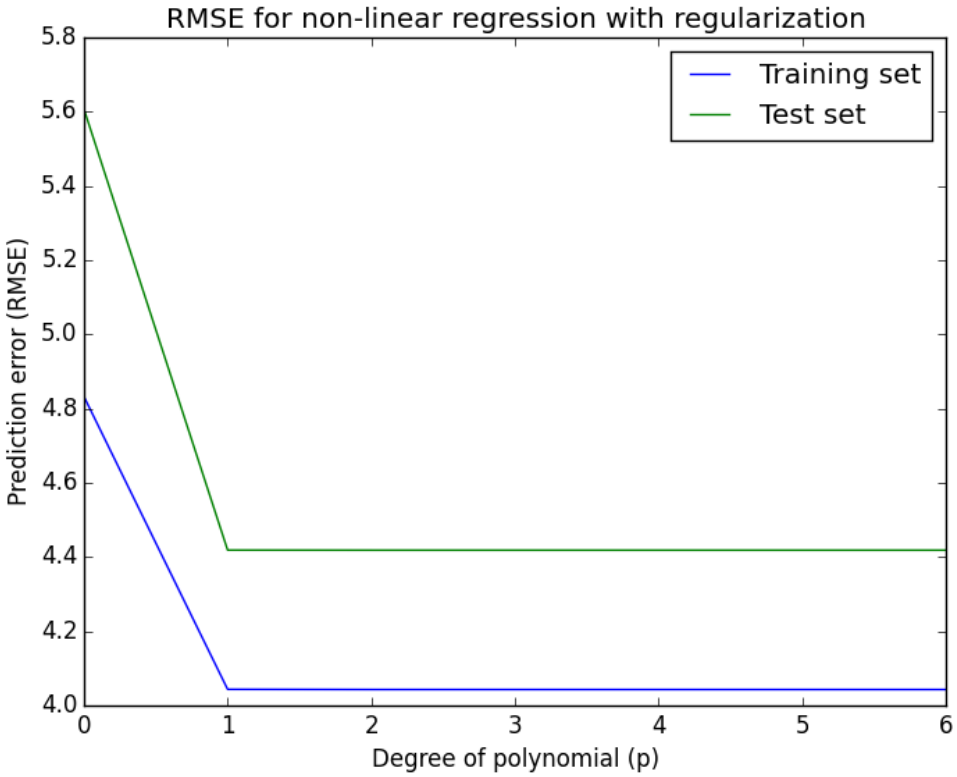

Figure 5.4: RMSE with regularization

Both RMSE decrease, but the gain is minimal, as depicted in figure 5.4. The minimum is reached at p=6 for the training set and p=5 for the test set.

In this case, the optimal value is p=5, but considering the minimal benefit and the increased computational effort in evaluating higher order polynomials, p=1 and in general all values from 1 to 6 seem reasonable choices.

6 Problem 6 - Interpreting Results

6.1 Comparison between the various approaches in terms of training and testing error

Problem	Training RMSE	Testing RMSE
2 - Intercept	3.0063021236	4.30571723465
2 - No Intercept	8.88388057487	23.1057743271
3 - Optimal	3.18062699378	3.7758233193
4 - Optimal	3.19157841599	3.7760118187
5 - Optimal (No Regularization)	3.99735628	4.38465206
5 - Optimal (Regularization)	4.04305081	4.4185436

Figure 6.1: Comparison between different approaches

6.2 What metric should be used to choose the best setting?

One clear metric for selecting the best algorithm is the testing error, since it shows how accurate the algorithm behaves for classification. According to this metric, we see no reason to recommend regression without using an intercept, since that gives a higher error for both training and test set. Linear regression gives the least training error, but when it comes to test error, Ridge regression excels. Non-linear regression does not perform as well as Ridge regression, that is the best approach in this scenario.

It is worth to notice that in some cases, like in problem 2 with linear regression and problem 5 with high degree polynomials, a decreasing training error may lead to think that is the best approach. Unfortunately, in many cases (and problem 5 is one of those) that only means overfitting, and the dramatic increase of the test error confirms it. Having a high test error is undesirable in any circumstance, and therefore, training error alone is not a good metric.

A further metric that is particularly significant in scenarios with limited resources is running time. In this case, and in general with relatively small data sets, it does not matter much, but when handling large data sets, runtime can be an important issue. In those scenarios, Ridge regression through matrix inversion may be unfeasible, whereas gradient descent could provide results faster at an acceptable loss of accuracy.

6.3 Summary and Conclusions

Summarizing, the results seem to indicate that non-gradient descent and gradient descent Ridge regression produce nearly identical results for the given data set, both of them producing the best results in terms of accuracy. If we take runtime into consideration, we noticed that non-gradient descent Ridge regression runs significantly faster for the given data set. In general,

gradient descent would be faster when dealing with huge matrices, while non-gradient descent Ridge regression would be faster for small data sets. It should also be noted that in general regularization will help produce better results as seen in problem 5.