Data Mining

Assignment1

Spyridon Roumpis

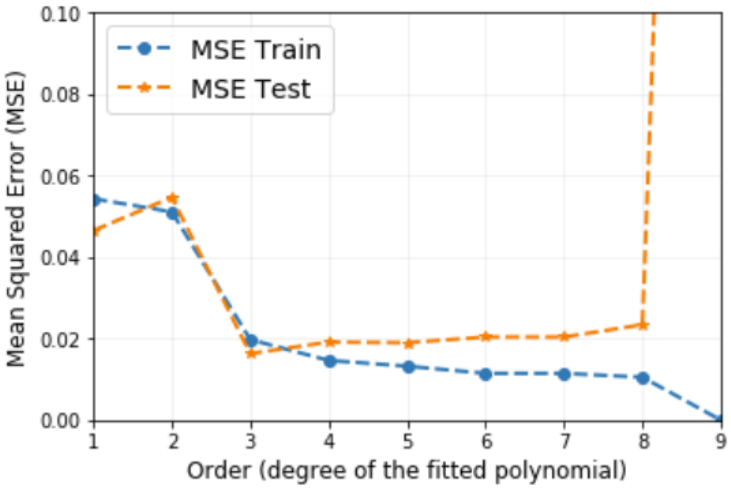
181004877

Exercise 0: Which polynomial has the lowest train MSE? Which one has the lowest test MSE?

For the purpose of this exercise we compare the Train MSE and the Test MSE for all of the 9 different degrees of the fitted polynomial in a summarized graph.

Fig 0: Comparison of Train and Test MSE (no regularisation).

You should be able to describe and explain the general trend in each graph, and identify the cases of underfitting and overfitting.



From the screenshot of the summarized graph we can see that the 9th degree of our polynomial has the lowest value of the Train MSE while for the Test MSE is the 3rd degree of the polynomial that has it.

Exercise 1: What trend do you observe when you analyse the dependence of train and test MSE on the polynomial order? First describe the observed trends, and then explain them.

From the summarized graph we can observe that Train MSE continually decreases until it reaches the value of zero for the 9th degree of the polynomial. This trend is something that we expect to happen, as we know from theory that as flexibility in a model increases (by increasing the polynomial degree) the training MSE keeps decreasing due to the increased flexibility. We also know that linear models are inflexible (1st degree polynomial) and that is the reason why it has the highest Train MSE.

However, a quite different trend can be seen in the Test MSE. The Test MSE decreases as we add flexibility up to a certain point, in our case this is happening at 3rd degree, although a small increase observed at the 2nd degree of the polynomial. After this point (3rd degree) the Test MSE continually increases and after the 8th degree is start becoming quite huge. Again, in this case we are expecting this trend as we know that after the Test MSE reaches the lowest point then it will keep increasing as long as the flexibility keeps increasing after this point.

Exercise 2: Identify the models that are suffering from under-fitting and the ones suffering from over-fitting. Justify your choice based on your observations and use the theory that we have learnt to explain it.

The models that are suffering from under-fitting, are the first two models (1st and 2nd degree polynomials). From theory we know that models which have both high Train and high Test MSE are models that suffer from under-fitting. In both of our cases, it can be seen in the summarized graph that Train and Test MSE are both high for the first two models, so they both under-fitting. Also, it can be seen, that both of these models have low variance and high bias, which means they suffer from under-fitting.

The models which are suffering from over-fitting are the models whose polynomial degree is 4 or higher. Theory tells us that an overfit model has extremely low Train MSE but a high Test MSE. In all of our over-fitting cases it can be seen in the graph that while the Train MSE is decreasing the Test MSE is increasing for the 4th till 9th degree of the polynomial. The same trend probably will still continue happening for degrees greater than 9. Also, it can be seen, that for all of these models the bias is low while the variance is quite high which allows the model to account more and more accurately for every single training point and it would work perfectly if there was no noise in the training data and our model did not end up fitting the noise.

Exercise 3: Which model would you pick as the best one amongst these 9 models? What are the parameter(s) and hyper-parameter(s) of your chosen model?

The best model is the model that has the lowest Test SME. From the graph it can be seen that the model with the lowest Test MSE is the 3rd model (3rd degree polynomial). Anything below this degree will underfit while anything above it will overfit. In order to get a good fit, we will take the point from just before where the error starts increasing, in our case for the 3rd degree polynomial.

Parameters are variables which are configured internally to the model and their value can be estimated from the data sets. In other words, the model uses them to make predictions. In our case the coefficients are a parameter.

Hyper-parameters are the ones that help with the learning process. They are configured externally, usually by hand, and they cannot be estimated from the data. In our model the degree of the polynomial is a hyper-parameter.

Exercise 5: Focus on order=9. Describe AND explain the trend of each of the metrics below with respect to increasing values of λ (that is, first describe what the effect of increasing λ from zero upward is it on the parameter in question and then explain briefly and clearly the reasons behind it): (a) TRAIN MSE

(b) $\vec{w}^T.\vec{w}$

(c) TEST MSE

To analyse the different metrics with respect to increasing values of λ, we ran the code and created a table with two columns. The left column has the different values of λ, while the right column shows the different metrics and how they are affected by λ. Also, for the purpose of this exercise we will focus only in the 9th degree polynomial.

lambda=0.0000	MSE Train = 0.0096 MSE Test = 0.0090 w'*w = 11.9345
lambda=0.0010	MSE Train = 0.0096 MSE Test = 0.0091 w'*w = 1.9366
lambda=0.0100	MSE Train = 0.0096 MSE Test = 0.0092 w'*w = 1.7026
lambda=0.1000	MSE Train = 0.0097 MSE Test = 0.0099 w'*w = 1.4674
lambda=1.0000	MSE Train = 0.0103 MSE Test = 0.0119 w'*w = 1.2599
lambda=10.0000	MSE Train = 0.0135 MSE Test = 0.0152 w'*w = 1.2047

(a) For the Train SME it can be seen that the value remains the same for the first three values of λ (λ =0, λ=0.001 and λ=0.01), MSE Train = 0.0096. For λ=0.1 there is tiny rise, the MSE Train = 0.0097. Furthermore, for λ values above 1 the figure of the MSE Train is increasing steadily, for λ=1 the Train MSE =0.0103 and for λ=10 the Train MSE =0.0135. From theory we know that while the λ is increasing the variance is reducing. For the three first values of λ the increase of λ is not that big hence the Train MSE does not really change.

(b) For the $\vec{w}^T.\vec{w}$ it can be seen that for λ=0 the value is 11.9345 and as the λ increasing the $\vec{w}^T.\vec{w}$ is decreasing. There is a huge drop in the figure when λ from 0 becomes 0.001 and then for the rest of the λ the drop of $\vec{w}^T.\vec{w}$ is more graduate. We know from theory that as the values of λ rises, it reduces the value of coefficients and thus reducing the variance, which explains the decreasing of $\vec{w}^T.\vec{w}$ in our model.

(c) As the values of λ rises the values of the Test MSE are decreasing steadily, starting from 0.009 when λ=0 and ending up at 0.0152 when λ=10. From theory, as λ increases, the complexity of the resulting solution decreases and so does the risk of overfitting, although after a certain value, the model will start loosing important properties, increase the bias and create an underfitting model.

In general, Regularization, reduces the variance of the model thus the risk of the model overfitting, so the parameter λ 'controls' the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance, something we already observed in our model.

Exercise 6: Now suppose that instead of 10 training instances, we had access to 100 train instances. Run the following script and inspect the change in the test error. Describe and explain the effect of having more training data on the test error (test MSE) and over-fitting.

	order=9	
lambda=0.0000	MSE Test =	0.0090
lambda=0.0010	MSE Test =	0.0091
lambda=0.0100	MSE Test =	0.0092
lambda=0.1000	MSE Test =	0.0099
lambda=1.0000	MSE Test =	0.0119
lambda=10.0000	MSE Test =	0.0152

To analyse the different values of the Test MSE with respect to increasing values of λ, we ran the code and create a table with two columns, the left column has the different values of λ, while the right column shows the different values of the Test MSE and how they are affecting from λ.

As the values of λ rises the values of the Test MSE are decreasing steadily, starting from 0.009 when λ=0 and ending up at 0.0152 when λ=10. From theory, as λ increases, the complexity of the resulting solution decreases and so does the risk of overfitting, although the figures haven't changed after running the script for 100 instances. Ideally, with more training data we would expect lower Test MSE, as the variance of the model would decrease and thus less chance of overfitting, but more data does not always mean more accurate model.

Exercise 7: Which is the correct way to complete the script and calculate the variable SSE? Explain what each of the three options to calculate SSE would do and justify your choice. What is the resulting train and test errors?

```
Lambda: 0.00000, CV SSE: 0.14715. * New best
Lambda: 0.00100, CV SSE: 0.28123.
Lambda: 0.01000, CV SSE: 0.26907.
Lambda: 0.10000, CV SSE: 0.13880. * New best
Lambda: 1.00000, CV SSE: 0.33508.
Lambda: 10.00000, CV SSE: 0.95373.
```

The correct way to calculate the variable SSE is by using the Val set:

SSE = np.sum(np.power(yvalset - np.matmul(phival, w_map), 2))

The script in the example is an already trained and tested model and we just need to validate it, so using again the same training and test set would not benefit us. Also, the Val set gives us the best SSE (screenshot).

From theory, the whole point of cross-validation is to test the already trained model (the model had already be given the training and the test set), ‘ train’ the trained model with a validation set of data, so the model can flag overfitting or bias selection problems. This is the main reason for choosing the Val set in our script.

Exercise 8: Compare the training, validation and the test errors in both the linear model and the M5P model. First, explain the differences between the numerical values of each error separately for the linear model and for the M5P model. Then, bearing in mind that the M5P model is more complex than the linear model, explain why the numerical values of the errors seem to behave differently for the lineal model and for the M5P model.

Below there is a presentation of the different metrics from the results of each error (training, validation and the test) for the Linear and the M5P models in WEKA.

Linear Model

- **Training set**

  Time taken to build model: 1.5 seconds

  Correlation coefficient 0.8923

  Mean absolute error 4229.4683

Root mean squared error 7357.4334

Relative absolute error 43.9307 %

Root relative squared error 45.1391 %

Total Number of Instances 9080

- **Cross Validation**

  Time taken to build model: 1.49 seconds

  Correlation coefficient 0.87

  Mean absolute error 4382.2685

  Root mean squared error 8049.8509

  Relative absolute error 45.5213 %

  Root relative squared error 49.3865 %

  Total Number of Instances 9080

  • **50% Test Split**

  Time taken to build model: 1.49 seconds

  Correlation coefficient 0.8734

  Mean absolute error 4434.7981

  Root mean squared error 8062.6951

  Relative absolute error 45.802 %

  Root relative squared error 48.9458 %

  Total Number of Instances 4540

Non Linear Regression – Trees M5P

- **Training Set**

  Number of Rules : 34

  Time taken to build model: 7.83 seconds

  Correlation coefficient 0.9351

  Mean absolute error 2968.1157

  Root mean squared error 5776.4447

Relative absolute error 30.8293 %

Root relative squared error 35.4395 %

Total Number of Instances 9080

- **Cross Validation**

  Number of Rules : 34

  Time taken to build model: 7.51 seconds

  Correlation coefficient 0.8983

  Mean absolute error 3453.2504

  Root mean squared error 7161.2058

  Relative absolute error 35.871 %

  Root relative squared error 43.9346 %

  Total Number of Instances 9080

- **50% Test Split**

  Number of Rules : 34

  Time taken to build model: 7.46 seconds

  Correlation coefficient 0.8929

  Mean absolute error 3469.4293

  Root mean squared error 7416.4842

  Relative absolute error 35.8318 %

  Root relative squared error 45.0229 %

  Total Number of Instances 4540

For the Linear Model we observe that the training set has the lowest error and the best correlation coefficient. The 50% test split set has the worst metrics and that is because of the number of instances, the model is using only half of them. For the cross validation we might would expect to have better results, although in our test we tried a 2-fold cross-validation which is significantly small k. If the k was at least 10 we would have a much better cross-validation model.

The same trend seems to follow the trees-M5P, the training set has the best correlation coefficient and the lowest error, followed by the cross-validation model and in the end is the 50% test split model. However, the differences in the values are not as big as they are in the linear model. For the same reasons, the small k and small test set the training set model is behaving better.

The M5p is a lot more complex than the linear regression model, that means Linear Models have very fewer parameters than M5P trees, so M5P may overfit more easily.