CM50264
10 January 2020

Regression algorithm comparison

Lab 5 - CM50264

Sections:

Word count: 2600

Section 1 Introduction

For the purpose of this task a dataset is being analysed which models a robotic arms movements. This dataset has 22 fields and contains roughly 44000 items - the intention is to model the function which maps 21 fields to the result position. This process of mapping the result data to learn the function is known as regression - for the purpose of this paper four different methods of regression are to be explored and compared against one and other. As such the methods shall be validated against a toy problem before compared in terms of computational performance prediction accuracy on the robotic arm test set.

Section 2 Method explanation

Section 2.1 Random Forests

Random forests are the first method being considered which warrant an explanation. In the case of a regression forest a combination between decision tree logic and ensemble learning is utilised to make inferences about the current class or value of the input data.

The first major idea as such in a random forest is the notion of a decision tree. A decision tree is a recursive method which loops through the data and decides on split points - based on criteria such as information gained or the mean standard error. From these splits the data is divided into bands which it can be classified or have a value attributed.

In the random forest technique multiple trees are used and trained, the key to this being performed successfully is a technique called bagging. Bagging is the process of producing random subsets of the original dataset - with replacement. Given enough trees the entire dataset will be predicted upon. The key is each tree trained on each dataset will have learned slightly differently and will contribute something different to the average.

This leads into the technique of ensemble learning. - which is the process of combining all of these trees each with different models of how to predict the values and combining the predicted value to produce a mean value. This allows for the best of all the decision trees features to contribute towards making a more accurate decision.

Regression forests as such are great with providing a function for discrete datasets. They are great for classification tasks due to the natural splitting nature of the tree leafs.

Another advantage of utilising random forests is because of the bagging, not a very large dataset is required as even with a small dataset splits and inferences can be made to provide sufficient training to trees.

One con of utilising random forests is they don't provide great value interpolation, it can predict a reasonable but not have means to provide a more accurate example than any provided in the dataset. Alternative methods such as the gaussian process are better for this value inference.

Section 2.2 Gaussian Process

A gaussian process is a distribution over a function - specifically a distribution which makes a covariance matrix from the data and the mean. The function can have an infinite domain and as such is useful for regression tasks as evaluating the mean and covariance at infinite points is not practical. As such the gaussian process makes use of a kernel function to evaluate our data. The kernel function is as such can provide priors from which we can create a posterior distribution. The posterior is obtained via bayesian statistics and using the laws of conditional probability to make inferences about the mean and covariance based off the assumed function in the kernel.

Gaussian process also has the advantage of being able to model confidence as by directly returning the covariance the standard deviation for each point can be obtained and used to observe accuracies at any given region.

The gaussian process has the advantage of being able to model functions to continuous datasets very easily which could in some models be an advantage as the way it produces a function to model your data could be necessary for that task.

The gaussian process also has the advantage of producing extremely accurate inference as it has incredibly high confidence around areas which it has existing knowledge. This is also reported in the deviation and can be considered when subsequently processing the data.

The major negative of gaussian process is the process of both designing and subsequently tweaking kernels to suit a given dataset with hyper parameter selection is very difficult. It is incredibly hard for a human to making predictions about which changes to a kernel will produce the most accurate function - as such great strides are being made in machine learning automation for kernel optimisation however it is not easy to do by hand still.

Section 3 Toy Problem

For the choice of toy problem a sample of data was chosen from the distribution of y = 2.5x with a slight bit of random noise included to ensure the algorithms could handle this but not so much as to spoil the simplicity of the data.

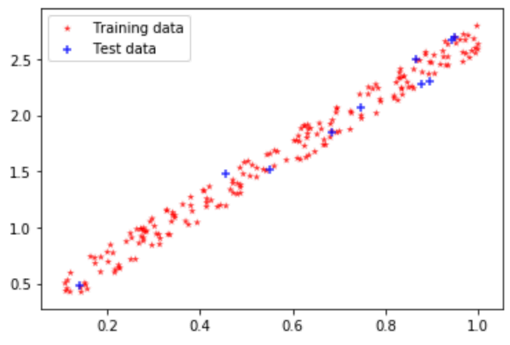

Figure 1 - Test and training data scatter plot

Section 3.1 Nearest Neighbour

The first of the regression algorithms tested was the nearest neighbour algorithm. This algorithm has the hyper parameter of k - which defines the number of neighbours which it shall cluster to. As such when completing this test a range of k values between 1 and 20 were tested to find out which was the most effective for RMSE.

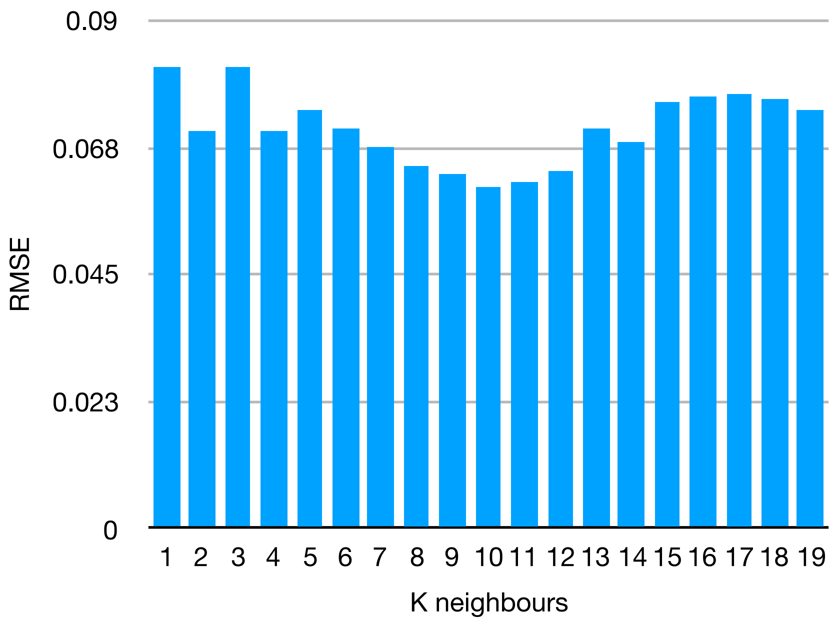

Figure 2 - Results of RMSE vs K neighbours

The dataset which was trained on was as state in section 3 a linear model (y = 2.5x) with minor noise - specifically the dataset 200 samples and 10 separate testing points which were then used to evaluate the performance of the algorithm. The testing points were separate as naturally if they were a part of the training set the neighbours would perform vastly better (as there would be a perfect match).
The final result of the test was that the average across the 10 testing items, with the optimal K neighbours value of 10, was an RMSE value of 0.0604741128878353. This was an exceptional score, but it was to be expected with a problem that has such low dimensionality and lots of nearby clustered points.

Section 3.2 Linear regression

The second of the algorithms tested was linear regression. Linear regression is a means of modelling all of the data to the equation y = mx + c. As this model is the model of y = 2.5x + 0 with minor noise padded along - the linear regression handled this problem fantastically.

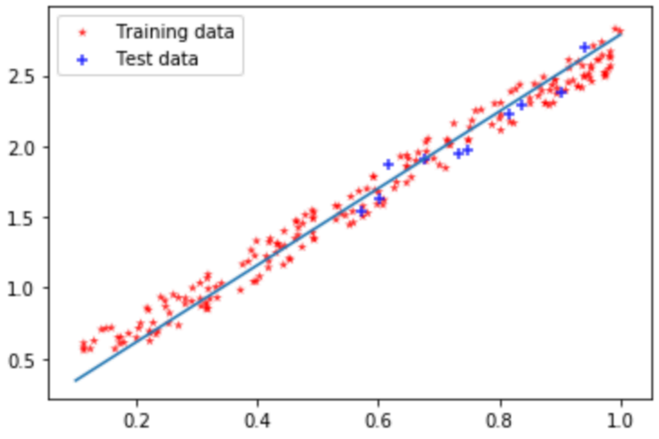

Figure 3 - Plot of linear regression versus test data

The model was trained as per every model with 200 values and subsequently assessed using a sample of 10 testing values. As linear regression has no hyper parameters to optimise there was no additional complications added to the program.

As can be seen though - when the line that the regression returns is applied into the formula y=mx+c the line perfectly bisects the data and points to the correct origin roughly. This is indicative of the algorithm working correctly. This is then further supported by the very good RMSE value obtained from the test set of 0.09601969051276356. This value being so low is very indicative that the model has correctly approximated the function of our set and as such will be able to handle the full dataset later on.

Section 3.3 Random Forest

The third of the algorithms to be trained was the random forest. Much like the predecessor algorithms the same toy problem was used making use of the same equation, noise and size of sample and test data. Unlike the predecessors however the random forest presented a larger amount of hyper parameters to optimise for. These included the number of trees and the depth of each tree.

The first parameter optimised for was the number of trees, this would assist in the ensemble learning as the more trees present the greater the variety in inputs we would have to the overall model. Unfortunately the more trees that are present will dramatically increase the computation cost of making the forest and potentially diminish the contributions of some of the trees to the overall ensemble. As such tests were performed to optimise the number of trees - using models of 3,5,10 and

15 trees respectively. Whilst 15 was marginally better than 10 it was within the margin for error and 10 was significantly faster to train as such was selected as the best candidate.
Going forward even further the values of max depth had to be considered. The max depth is the amount of splits that can occur in a branch before termination is forced and an estimation is made. For the experiment with max depth three depths were considered 3,5 and 15. Once more this replicates the results observed with regards to the number of trees with regards to 15 winning marginally but due to better computational performance and the margin of error separation the max depth of 5 was selected as the optimal max depth.
With these optimised hyper parameters in mind the normal test of 10 items with 200 training items was subsequently performed with a max depth of 5 and 10 trees in the forest. This resulted in an RMSE value of 0.11599260901294367. Which is fairly low, it certainly shows that the regression forest is performing correctly and can be used on the main dataset.

Section 3.4 Gaussian Process

The final of the algorithms is the gaussian process. The same initial function of y=2.5x+0 was used for the initial dataset - produced with the usual noise and 200 items to sample alongside 10 items to test on. With regards to kernel optimisation - the rule of thumb was as long as the bounded region denoting the deviation from the mean roughly matched the shape of the test set the kernel was fairly optimal.

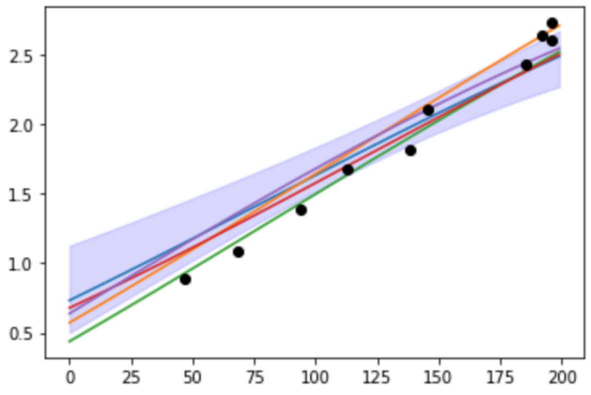

Figure 4 - Functions from gaussian graphed against test data

From this a total of 5 functions were defined from the posterior. These functions are all drawn on the graph versus the test data - as seen they do a reasonable job of representing the data. Overall the gaussian process performs very well with the RMSE for one of the functions being only 0.18588361997811956 it's clear the method is working correctly. Further the method can be taken further and used in the main batch of tests.

Section 4 Experiments and analysis

All data in this section is making use of the SCAROS dataset. This section intends to use this data set to make more conclusions about performance features of each of the algorithms.

Section 4.1 Selection strategies

A hyper parameter is a value which is set prior to running the model which contributes to the performance of the model. Throughout the toy problem many hyper parameters were explored. These were once more repeated but with slight variation depending on the algorithm at question. With regards to linear regression, there was obviously no selection strategy employed as there were no hyper parameters.

As for nearest neighbours a subset of 5,10 and 15 were trialed for K on the new full dataset. The full 20 was not trailed once more due to the time to complete a cycle being too long with the new dataset. As such 10 was found to be optimal hyper parameter and was the value carried forward for all further comparisons of the algorithm.

For the random forest the full trials could not be completed due to time limits. As such only 5,10,15 trees were trialed and only depths of 5,10,15 were trialed. Ultimately this leads to a 10 max depth and 10 trees per forest being used.

With regards to Gaussian there were minor tweaks made by adjusting parameters by trial and error to experiment with what would reduce the RMSE the most. As stated previously tweaking the kernel to minimise the RMSE is very hard to visualise and would be better suited to a machine learning algorithm controlling.

Section 4.2 Performance Comparison

The first thing to note is now this data set is considerably larger, with 22 dimensions of data and 44000 fields this is a much larger problem than the 2 dimensions with 200 items from the toy problem. As such some of the solutions had to have limited data sets to complete the problem in reasonable times. The first example is the regression forest which could only have 4000 items to train on (which was sufficient) and only 100 items to test on. Had there been any more the time to completion would have been impractical to allow for proper tuning.

The second algorithm which had to be limited was the gaussian process which had to be limited to only 1000 records and 100 in the test set. Had this not been limited time would also have been a significant issue.

Finally both linear regression and nearest neighbours have been limited to only 20,000 records and 100 test items for the purpose of the following tests.

The first test is the time to complete a test and report the RMSE for the given dataset. This gives a rough estimate on the computational complexity of the solution.

Computation time:

1	Linear regression	5 seconds
2	Gaussian process	30 seconds
3	Nearest Neighbour	2 minutes 30
4	Random forest	12 minutes

The second test subsequently looks at the performance in terms of RMSE. This means looking at the predictions versus the test set and reporting the square root of the error.

RMSE:

1	Linear Regression	21.972021039102803
2	Random Forest	27.40938279953293
3	Nearest Neighbour	33.19317016968418
4	Gaussian Process	35.69717026631414

Section 4.3 Analysis and evaluation

Surprisingly linear regression was the best performing of all the algorithms in terms of error. Whilst the error was still moderately high it infers that there was a very linear relationship in many dimensions which was able to be identified and subsequently modelled quite easily.

Following on from this the random forest did fairly well, given however the significant training time provided it does seem to have not performed as well as expected. This was however likely due to even when cut down how much data was there to be processed - which was likely still too much when techniques such as bagging were happening. However the algorithm did still work and did return reasonable data still. Perhaps despite optimal hyper-parameters being used for the RMSE value a more optimal value could be found to reduce training time.

One of the biggest surprises in this was the poor performance of the Gaussian Process, it could be the variety in dimensions hindered the process as it doesn't handle large amounts of dimensionality very well or it could just be a poor kernel which could have been configured better. It also could be an indication that noise was oversampled for in the model, or perhaps under sampled, throwing the model off.

Nearest neighbour meanwhile was middling, it obviously got enough data with sufficient similarities that it could function but clearly either not enough data or there was enough variance that the estimations were quite off.