Random model trees: a competitive off-the-shelf technology for near infrared spectroscopy

B. Pfahringer, D. Fletcher, R. Bouckaert and G. Holmes
University of Waikato, Hamilton, New Zealand.
E-mail: bernhard@cs.waikato.ac.nz

Introduction

The standard prediction method for NIR data is PLS regression which works very well for linear and almost linear relationships. One disadvantage of PLS regression is its inability of modelling non-linear relationships. Universal approximation algorithms, such as support vector machines, neural networks or Gaussian Processes, can fit arbitrary non-linear relationships, but that ability comes at a cost: usually multiple parameters have to be tuned for good predictive performance and computational complexity as well as the fact that actual run-times can be problematic. This paper focuses on an alternative technique for modelling non-linear relationships: model trees combine local linear models using a decision tree that splits the data into disjoint regions. The approach presented here randomizes this algorithm and generates ensembles that consist of multiple such model trees. Similar to the approaches listed above, the method allows for tuning a number of parameters. These include the regularization value for each local linear model, the depth of each tree, the number of features to consider at each split node in the tree, as well as the total number of trees. The main contribution here is the fact that a set of sensible default values for all these parameters works very well across a range of different NIR datasets. Another advantage of this approach is its scalability: both training time and model sizes are O(NlogN), where N is the number of training examples.

Materials and methods

Model trees are grown by recursively splitting the data into subsets according to an automatically determined threshold for one of the features. The selection of these features is randomized: a small number of all features is chosen at random, and then that feature where the split based on the median value for that feature which minimizes the *RMSE* over both subsets, is selected. Trees are grown up to specified depth, then at each leaf a linear regression model is computed for the respective subset of the training data. The default values for parameters are set as follows: 250 trees are grown for an ensemble; only three percent of the features are considered in every random split selection; the depth of the tree is a function of the number of training examples: as each split roughly halves the size of each subset of examples, splitting is performed until the number of

examples is less than two times the number of features. For small datasets that could lead to trees of size one, or to even one global model, therefore a strict minimum tree depth of two is enforced to guarantee appropriate diversity of all trees. The ridge value for the local regression models is set between 0.001 and 0.0001, depending on tree depth. Experiments using seven real world NIR data sets are produced below. Each spectrum consists of 171 features after preprocessing the raw data using Savitzky-Golay filtering.

Results and discussion

Table 1 shows an overview of the RMSE for linear regression on PLS filtered NIR data where the optimal number of PLS components was determined for each of the seven datasets separately, and for Random Model Trees (RMT) on NIR data with 171 attributes. The differences are statistically significant at 5% for all data sets except for the two smallest one (Lactic and Storig), using the corrected paired t-test. These experiments show that RMTs without any parameter tuning perform very well over a wide range of data set sizes. The training times show the expected O(NlogN) behaviour, e.g. dataset N is 7.5 times larger than dataset OMD, and it needs about 15 times as long to train. Model sizes and prediction times have been omitted here for space reasons, but follow the same trend.

NIR data description

In this paper we describe a generic regression method that according to Table 1 can consistently outperform the standard method used for predicting from NIR spectrograms, namely PLS regression.

This is true even if PLS was specifically optimized for each single application, and when only appropriate default values for the various parameters were used for the new method. All

Table 1. All results based on 10 times repeated 10 fold cross validation. Number in brackets is standard deviation.

Data set	Size	Root mean square error		RMT train time (seconds)
		PLS regression using optimal number of components	Random Model Trees	
Lactic	255	0.469 (0.074)	0.461 (0.082)	0.584
Storig	414	2.037 (0.385)	1.776 (0.664)	1.183
SS	895	1.316 (0.136)	1.011 (0.146)	5.528
OMD	1010	3.152 (0.497)	2.766 (0.541)	7.310
DCAD	2522	70.822 (8.887)	59.865 (6.298)	30.345
K	6363	0.366 (0.022)	0.247 (0.026)	86.208
N	7500	0.212 (0.022)	0.155 (0.028)	100.245

datasets were produced with an NIR machine that generates spectrograms comprising roughly 700 raw values. A Savitzky-Golay filter using a window size of 15 was used to both smooth these values and to reduce the total number to only 171. This filtering also automatically removes any effects due to base line shifts in the NIR machine. Down-sampling to 171 attributes seems to preserve all necessary information and speeds up computing linear regression models tremendously, as the usual solvers for linear models are of a computational complexity that is cubic in the number of attributes. Reducing the number of attributes by a factor of 4 can speed up such a solver by a factor of 64. The default values for all necessary parameters of the algorithm described below will work well for any problem pre-processed in this way. When using different machines or a different form of pre-processing, new sets of reasonable default values would need to be determined.

Random Model Tree Algorithm

On small data samples the main effect to predict is usually well modelled by a global linear regression model, even if the effect in itself is not completely linear. Only when more data becomes available can non-linearity be extracted in a reliable way. Samples sizes of 500 or even several thousand samples might be necessary. Gaussian Process Regression[1] is a very well-performing non-linear method for such problems, especially when using so-called Radial Basis functions as a kernel. A major issue with Gaussian Processes is their computational complexity: their memory requirements grow quadratically in the number of samples, and the time complexity grows even cubically in the number of samples. Random model trees are a more scalable alternative, as their complexity is only O(NlogN).

Random Model Trees are the combination of two existing algorithms in Machine Learning: single model trees[2] are combined with Random Forest ideas.[3] Model trees are decision trees where every single leaf holds a linear model, which is optimised for the local subspace described by this leaf. This works well in practise, because piece-wise linear regression can approximate arbitrary functions as long as the single pieces are small enough. For differentiable functions it can also be viewed as a crude one-step Taylor series expansion of such a function. Random Forests have shown that the performance of a single decision tree can be improved by the addition of randomization and by averaging multiple such randomized trees. In a Random Forest every single tree is grown from a bootstrap sample of the training data, and each tree is grown by not always splitting on the best possible test, but instead by splitting on the best test out of a small randomly drawn sample of all possible split tests. To the best of our knowledge no-one so far has combined these two ideas into Random Model Trees for regression.

The success of Random Model Trees critically depends on some specific engineering features. When an attribute is considered for splitting the data, then only the median of the values of that attribute in the current subset is considered as a split point: this guarantees roughly equally sized subsets. Thus the generated trees will be roughly balanced and all subsets at each leaf will be of reasonably similar size, which allows for the use of the same globally specified parameter setting for each local linear regression. Whenever the number of samples at a leaf is too small, then a subset of all attributes is selected to ensure that the number of samples is at least twice the number of attributes. A simple linear-complexity attribute selection method seems to work well in practise: attributes are ranked by their correlation with the

target attribute, and only the top-K attributes are selected from that ranking. Additionally, to prevent against extreme cases of extrapolation, the extreme values for the target are recorded in each leaf. If the prediction from a linear model lies outside these extreme values, the prediction will be capped back to the respective extreme value. Finally, as the trees are semi-random and therefore definitely not optimal in isolation, averaging an appropriate number of such trees is essential for good predictive performance. At least 30 trees should always be computed, and computing a lot more can sometimes further improve performance. Because of the random nature of the process adding more trees will never degrade performance, but as with most ensemble methods improvements diminish eventually.

For experiments reported in this paper 250 trees were generated for each problem. The height of any tree was set such that it was at least 2 to ensure reasonable diversity, but limited such that each subset at a leaf would contain about 340 examples, i.e. twice the number of attributes. Given that trees are least two levels deep, this latter constraint is of course violated for all datasets with less than 1360 examples (four of the seven datasets in Table 1). The number of attributes considered at each split-point in the tree was set to five (about 3% of the total of 171 features). Experiments showed that performance is not very sensitive to specific values of this parameter, as long as it is not one (which would lead to completely random trees) and as long as it is also less than the square root of the number attributes (otherwise trees would become too uniform, and averaging consequently be meaningless). In the leaves regression models are computed by ridge linear regression, with a ridge value of 1.0E-4, or 1.0E-3 when tree depth is only 2.

Experiments

Results reported in Table 1 are averages of ten-fold cross-validation repeated ten times. Such averages provide a good measure of both the average performance as well as of the variance of a method. The latter is especially important to measure for randomized methods. Furthermore it allows to compute the statistical significance of differences between methods. In Table 1 all such differences are significant at the 5% level for the five largest datasets.

Regarding speed, even an ensemble for the largest dataset can be computed in about 100 seconds on a standard laptop. When analysing runtimes for all dataset sizes the measured times actually behave in a linear fashion, i.e. better than the expected O(NlogN) behaviour is observed. The best explanation is that for datasets of such sizes, computing the linear models, which is linear in the number of examples, still dominates the tree construction time which would be O(NlogN). For considerably larger datasets we would expect to observe non-linear increases in runtime.

Availability

An open source version of the algorithm will be made available as part of Weka[4] in 2010.

References

1. C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, Massachusetts, USA (2006).
2. J.R. Quinlan, "Learning with continuous classes", *Proceedings Australian Joint Conference on Artificial Intelligence*, pp. 343–348 (1992).
3. L. Breiman, "Random Forests, *Mach. Learn*. **45(1)**, 5–32 (2001).
4. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, "The WEKA Data Mining Software: An Update; *SIGKDD Explorations* **11(1)**, (2009).