Down and Dirty with Data

Introduction

The purpose of this project is to predict five soil properties from spectral data and other features as part of the Africa Soil Property Prediction Kaggle Challenge. The method will be used to predict the viability of land using a low-cost measurement technique.

Dataset

The five target variables are Calcium (Ca), Phosphorus (P), pH, SOC (Soil Organic Carbon) and sand. The training set consists of 3,595 features and 1,157 training examples, which are labeled with a conventional chemical soil test. Nearly all (3,578) of the features are IR spectral absorbance by wavelength. All values are continuous except for one feature: the topsoil/subsoil indicator. The test set contains 727 examples.

Features and Pre-Processing

Feature selection

The large number of features makes the data prone to overfitting and computationally challenging. We used filter feature selection and forward search to find a subset of features that are most relevant.

For filter feature selection, we calculated the Pearson correlation coefficient of each feature against each of the five target variables, and then highlighted the top 5% of positively and negatively correlated features. For each identified spectral band, we selected the most highly correlated feature and 1-2 features on each side of this peak to carry information about the width of the band.

Forward search was used with k-fold cross-validation and a linear regression model with the number of features determined by minimum test error (Figure 1). However, forward search is dependent on the order that features are selected. For example, we obtained lower test error when we initialized the first two features to be the highest positive and lowest negative Pearson-correlated features. Unfortunately, the best results from forward search did not exceed those of filter feature selection.

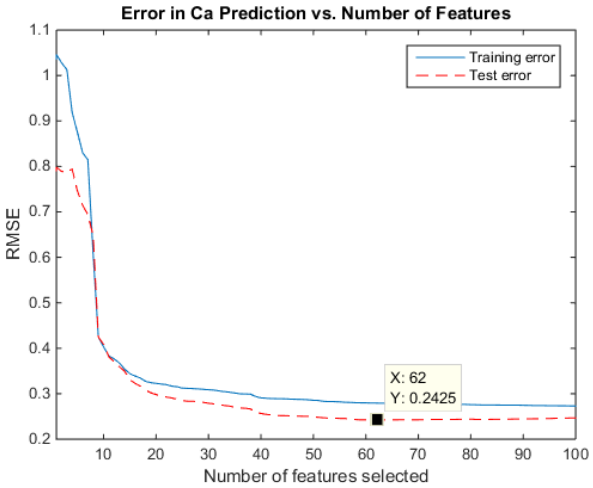

Figure 1. Forward search performed on Ca using linear regression.

Pre-processing

The first and second derivative of the frequency spectra were taken to perform baseline corrections (Chen and Wang). Peaks in the transformed spectrum data (Figure 2), which are the compressed encoding of the original spectra, were then put through forward search (Villmann et al). Principal component analysis on the transformed spectral peaks was also used to compress the data (Chen and Wang).

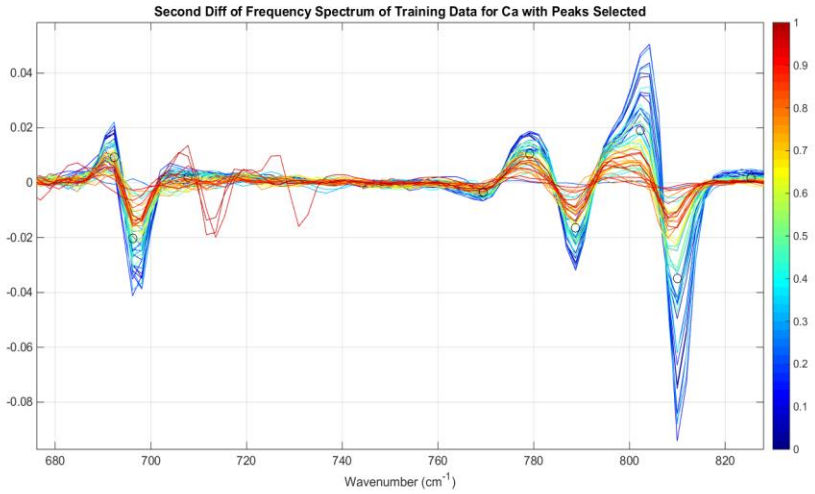

Figure 2. Second derivative of frequency spectra with peaks selected (shown with circles). The colors correspond to the value of Calcium for each training example.

Models and Results

Error metric

Mean columnwise root mean square error is the metric used to measure performance on all five of the target variables. The metric is simply the average of the root mean squared error of the five targets.

$$MCMRSE = \frac{1}{5}\sum_{j=1}^{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2}$$

Linear regression

Our first submission for Kaggle scoring used the features selected via filter feature selection. We used linear regression, because chemical concentrations are proportional to absorbance spectra according to Beer's Law (Chen and Wang). The Kaggle score was 0.61785 which would have ranked us 921 of the 1233 teams that competed. For comparison, the winning score was 0.46892, and the all-zero benchmark has a score of 0.91393.

Weighted linear regression

Next, we used weighted linear regression to improve upon the prior results with a single value of the bandwidth parameter, τ, for all target variables. The feature set was unchanged. Weighted linear regression improved results with an optimal value of τ = 1.5 (Table 1).

Bandwidth (τ)	Training Error	Test Error (Kaggle Score)
0.4	0.16319	0.92643
0.8	0.34775	0.63724
1.5	0.45806	0.59216

Table 1. Results of weighted linear regression for varying τ values

Optimization of bandwidth parameter

Next, we optimized τ in our weighted linear regression for each individual target. We optimized first using the training set, and then further tuned the value of τ for some target variables using the test set on Kaggle. The optimal τ values by target were Ca: 1.25, P: 1.5, pH: 1.14, SOC: 1.33, and Sand: 0.44.

From plots, it seems that at least two of the target variables (P and Ca) may have power law or other highly nonlinear distributions. In addition, target variables may be better predicted by some polynomial power of a set of features. To address these issues, we shifted the values of the training set data by 5 to remove the negative values, and performed forward search feature selection using ln(x+5) to predict ln(y+5) with linear regression. This produced the best linear regression model with a training error of 0.44267 and test error or 0.54918. Our best weighted linear regression model used the optimized τ for Ca and Sand and then re-optimized bandwidth parameters for P, pH and SOC based on the log transformation just described. This ensemble model had a training error of 0.4061 and test error of 0.54165. The training and test errors for all linear regression models are summarized with all other results in Figure 3.

Other regression methods

Linear regression was performed on the peaks of data that had undergone first and second derivative transformations. The first derivative transformation performed better than the second derivative (Figure 3). Its training error was smaller than linear regression on the untransformed data, but the test error was similar indicating overfitting (Figure 3) However, forward search performed on the peaks led to a larger test error than using all the peaks. Principal Component Regression (PCR) on the transformed derivative data performed poorly (Test error = 0.866), because the principal components of the features may not explain the variation in the target variables. Therefore, we used Partial Least Squares regression (PLS), which finds the directions in the feature space that best explains the variance in the target variables (Chen and Wang). PLS (Test error = 0.713) outperformed PCR, but it did not beat linear regression.

Regression tree models

We implemented regression trees for each target variable, allowing MATLAB to select any features from the complete set. The corresponding test error results were poor indicating overfitting by the full trees, but there was not time to optimize the pruning. However, we did use the decision tree results to inform whether to split the training set (and models) based on the one classification feature (topsoil/subsoil). One decision tree (SOC) showed significant dependence on this feature, so a dual model for SOC was constructed: one to predict SOC for subsoil examples and another for topsoil. Forward feature selection was used for each sub-model and the log transform was used with linear regression for prediction. However, the more detailed model for SOC did not improve upon prior results.

Neural network models

We implemented a feed-forward model for a neural network as a simple machine learning algorithm to reduce error and model bias (Madden and Ryder). We created a neural network for each target variable and trained it using the Levenberg-Marquardt algorithm. Table 2 highlights some of the many experiments we ran to determine the optimal parameters. We experimented training the networks using all the features, first and second derivatives of the spectral data, only the Pearson-correlated features discussed above, and other feature subsets that included handpicked spectral bands. The intent was to find a minimal set of features that performed well so we could reduce the degrees of freedom and avoid a local optima. In the end, our best result was trained using all features. We also experimented with the number of neurons and the transfer function used for each target's neural network (these results are omitted). The minimum training set error proved not to directly indicate the optimal neuron and transfer function parameters for the test set. The neural networks yielded the minimum error for Phosphorus, our problem target, which is why it was more successful than the linear models.

Neural network & weighted linear regression ensemble

We combined the neural network model for Phosphorus with the weighted linear regression models of the other targets. This yielded our best test error at 0.52578 and Kaggle rank 595.

Experiment	Neurons per Target	Transfer Function	Features per Target	Test MCRMSE
Tune Ca to 1 neuron	[1,2,2,2,2]	tansig	3594	0.529
Band reduced 2 neurons	[2,2,2,2,2]	tansig	1522	0.545
Fine band reduced	[2,2,2,2,2]	tansig	1863	0.624
Tune to best <6 neurons	[1,5,1,2,2,2]	tansig	3594	0.629
Band reduced 10	[10,10,10,10,10]	tansig	1522	0.707
Pearson feat. custom neurons	[2,27,50,18,2]	tansig	~30-40	0.734
Logsig transfer function	[1,2,2,2,2]	logsig	3594	0.749
First derivative features	[1,2,2,2,2]	tansig	3593	0.756

Table 2. Results of selected neural network models.

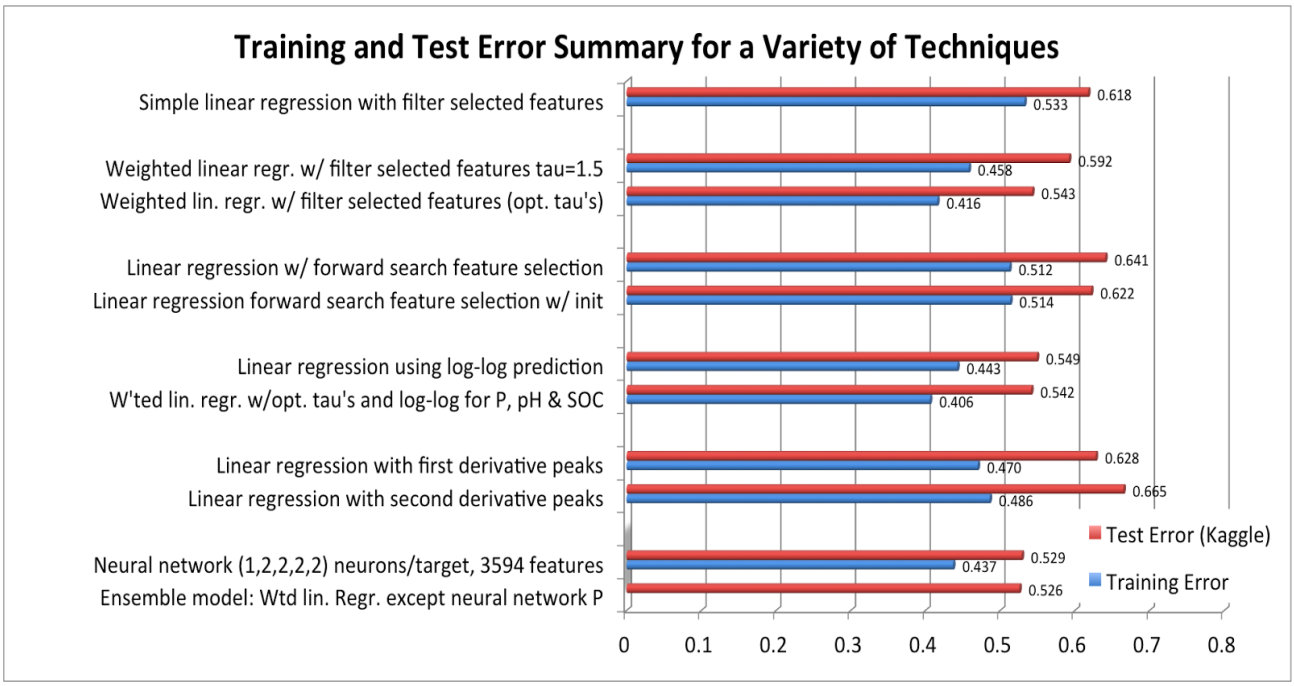

Figure 3. Model and feature selection comparisons for the project overall.

Discussion

The linear regression models were able to predict all of the target variables with a RMSE below 0.4 except for Phosphorus (P), which has a RMSE of approximately 1 (Figure 4). The training and test error are similar for P which indicates a bias problem. Phosphorus in the training set has a small number of high values that dominate the error, since the error is squared in the RMSE calculation. Other Kaggle teams achieved better results with linear regression models by removing these high values from the training data. Neural networks predicted P better than other methods, because P may be a nonlinear function of the given features. The improvement of the log transformation further supports this hypothesis. For the other output variables, the test error exceeds the training error by less than 0.08. This indicates a slight bias, which can be corrected by decreasing the number of features.

Filter feature selection performed better than forward search, likely because forward search tends to find local optima. Forward search is dependent on the order that features are selected. For the first derivative peaks, the test error actually increased when using forward search instead of all the peaks. In filter feature selection, the simplification of the spectral bands to a highly correlated feature and two other features decreased the number of redundant features to reduce overfitting. Even though forward search used k-fold cross-validation, forward search still overfit the data more. For simple linear regression, the difference in test and training error was 0.13 for forward search versus 0.09 for filter feature selection.

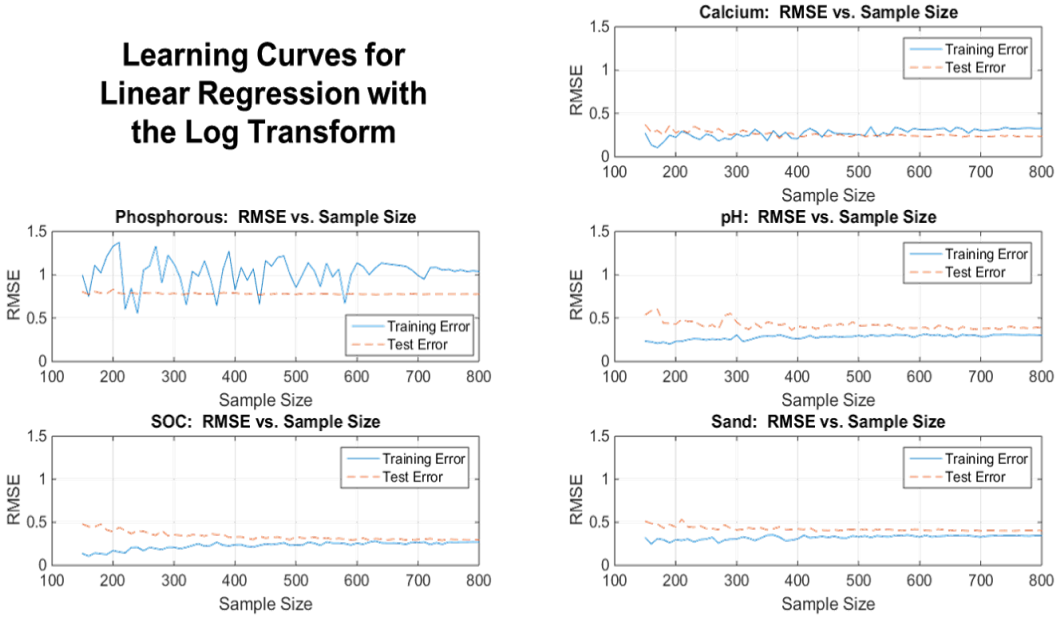

Figure 4. Learning curves for linear regression with the log transform ln(x+5) → ln(y+5).

Conclusions and Future Work

We used more than four techniques on this problem and performed well against the Kaggle competition on a problem of significant human importance. If we had another six months to explore this project, we would continue the initial work done with regression trees. Although we did implement the MATLAB regression trees for all target variables, we did not have time to optimally prune the regression trees. We would also improve feature selection with a genetic algorithm to learn the most significant features by avoiding local optima. Last, we would use a broader ensemble of methods as the winner of the Kaggle competition did.

References

J. Chen and X. Wang, "A New Approach to Near-Infrared Spectral Data Analysis Using Independent Component Analysis," *J. Chem. Inf. Comput. Sci,* vol. 41, pp. 992-1001, 2001.

H. A. Chipman, E. I. George, R. E. McCulloch (June, 2008). *BART: Bayesian Additive Regression Trees*. Available: http://www-old.newton.ac.uk/preprints/NI09002.pdf.

M. Madden and A. Ryder, “Machine Learning Methods for Quantitative Analysis of Raman Spectroscopy Data,” *Opto-Ireland,* vol. 4876, pp. 1130-1139, 2002.

T.M. Mitchell, “Decision Tree Learning” and “Artificial Neural Networks,” in *Machine Learning*, McGraw-Hill, 1997, pp. 52-127.

T. Villmann, E. Merenyi and U. Seiffert, "Machine Learning Approaches and Pattern Recognition for Spectral Data," in *The European Symposium on Artificial Neural Networks*, Bruges, 2008.