SUB-SAHARAN SOIL QUALITY MODELING

ERIC XIAO AND MARTIN ZHU

Abstract. Advances in rapid, low cost analysis of soil samples using infrared spectroscopy, georeferencing of soil samples, and greater availability of earth remote sensing data provide new opportunities for predicting soil functional properties at unsampled locations. Soil functional properties are those properties related to a soils capacity to support essential ecosystem services such as primary productivity, nutrient and water retention, and resistance to soil erosion. Digital mapping of soil functional properties, especially in data sparse regions such as Africa, is important for planning sustainable agricultural intensification and natural resources management.[1]

The Africa Soil Information Service has provided us with 1157 samples and 3594 variables from each sample, including 5 target properties: soil organic carbon, pH value, Mehlich-3 extractable calcium, Mehlich-3 extractable phosphorus and sand content. Using techniques from data science and machine learning, we will create a model to predict the 5 target properties of 728 other samples with the given 3594 variables from each of those samples using `MATLAB`, a powerful programming language for computation; we entered a competition where we are trying to create the most accurate model possible. The African Soil Information Service will use the models of the winning entries on their database of 19200 other soil samples to support high economic, social, and environmental returns on investments in agriculture that will improve the lives and livelihoods of farming communities in Africa. Their research affects 90% of the people living in Africa. [1]

Introduction

Fifty five percent of the land in Africa is useless agricultural-wise for anything but periodic grazing; most of this land is desert, rocks, and extremely steep land [3]. There has not been the expected 4-5% increase in the annual productivity of agricultural growth in relation to the immense amount of work put into development and research, and this dismal lack of improvement is slowly starving the 400 million people who depend on the 9 million square kilometers of developable land for sustenance. The low productivity of land is rooted in poor soil quality, as 5 million hectares of land in Africa has been degraded or overworked so much that restoring it would not be worth the cost [3].

Soil quality is measured by its ability to allow life to flourish, provide an ecosystem and filter water, and keeping an accurate status of soil quality will allow conservationists to know where to target programs, measure sustainability, and identifies risk zones where ecosystems are on the verge of collapse. Risk zones are define as land that is currently farmable but may become not arable in the near future if mismanaged. Zones with very high risk are in danger of becoming desert and are usually located along the borders of present deserts, while zones with low risk are good for sustainable farming and can be used for long into the future if managed correctly. There are currently around 22% of land, or 6.7 million square kilometers,

that is low risk, and an additional 4 million square kilometers can be classified as such if their use were more regulated. Unfortunately, there is around 54% of land, or 16.6 million square kilometers, that is at high risk, and 8.2%, or 0.7 million square kilometers, of land that is at extremely high risk at the edges of deserts [3]. Fortunately, it is estimated that some small technical changes, such as providing fertilizer to women, can bring as much as 43% of the land into low risk zones [3].

We know that development in Africa cannot follow the high input high yield models of other countries, and they do not have the technology or the resources to do so sustainably. The approach to soil management must be more fundamental and holistic. Africa has enough arable land to feed more than its population, and with proper management and detection this land can be expanded.

The Africa Soil Information Service (AfSIS)[1], the organization that created the competition, has the goal of improving the way that soil is mapped, monitored and evaluated as to solve some of the problems mentioned above. They wish to reduce poverty and promote sustainable growth and use of soil to prevent further soil degradation, as the soil regulates the hydrological cycle which is imperative to life everywhere. They plan to achieve this through improved data collection and better analysis. They also provide services such as technical training on soil sampling and spectral data management in order to better spread awareness and knowledge of sustainable soil use. This competition will allow AfSIS to develop new data algorithms and give them a much cheaper and faster way to collect reliable data on soil.

The Competition. The goal of the competition, details of which can be found at `http://www.kaggle.com/c/afsis-soil-properties`, was to predict physical and chemical properties of soil based on satellite data, providing a low cost way for scientists to analyze the quality of soil to help conservation efforts. The test, diffuse reflectance infrared spectroscopy, measures the infrared light absorbency of the soil, and can be performed in a matter of seconds, as opposed to traditional methods. We were given a test set of data containing thousands of points on the absorbency of different spectra of light from different patches of soil as well as the target values we had to predict, which were soil organic carbon (SOC), pH, Calcium (Ca), Phosphorus (P), and sand. We could use this data to create and refine our model, on which we would run another set of "real" data that was our submission.

Our results were judged on mean columnwise root mean square error, described by the equation

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

where y is the actual value from our results and $\hat{y}$ is the expected value, which we do not know. We are given a score on our data after every submission, and we are only allowed three submissions each day.

1. Data

As we mentioned earlier, we had two sets of data given to us, presented in CSV spreadsheet format. The training set consists of 1157 soil samples, 3594 points of infrared spectroscopy data as well as other points of data that may be relevant

[1] `http://africasoils.net`

to soil quality such as Black Sky Albedo and compound topological index. The training set also contains the actual values of the five target values that we are trying to predict. In the description of the data, we were told that the 15 carbon values of the 3594 variables could be omitted as they are probably irrelevant. Thus, we used 3579 variables to predict the 5 target variables.

The other set of data given to us was the test data that we had to run our model on and submit to the competition. This set contains 727 soil samples, with the same number of data points for each sample as the training data.

2. Approach

When we discovered and subsequently decided to enter this competition, we already had a very rough idea of how we would proceed. We knew that this competition dealt with big data and that we would have to use some form of machine learning, as most conventional big data projects find patterns in immense sets of data with some clever machine learning algorithms. We could see at once by just plotting the data that a simple linear regression would not suffice, as much of the data is non-linear. Thus, we knew we had to present our data in a way so that a machine learning program could do work on it and give us some meaningful results. Fortunately, MATLAB has some built in machine learning apps [2] that we could build off of.

2.1. **Machine Learning.**

Machine learning employs neural networks to handle pattern recognition and model building. Neural networks are an information processing tool that is built to resemble the human brain in problem solving and pattern recognition. Neural networks consist of many interconnected "neurons" that work in tandem to solve a problem [6]; in our case, the neural net is trying to find an accurate model that fits the training data. Unlike an ordinary computer program that uses an algorithm to solve a problem, neural nets do not have a specific set of instructions that it has to follow; instead, the network has a "mind" of its own and seeks to solve the problem through repetition and examples. An advantage of this approach is that it can solve very complex problems that may be hard to model, however, because there is no explicit instruction, neural networks may act unpredictably [6].

There are two types of learning that a neural network can do: supervised learning and unsupervised learning. Supervised learning is the process where neural nets are given a set of data and told what the results of that set of data should be so that it can correct itself and build better models, while unsupervised learning is where there are no results to give to the neural net and it must seek out patterns in the data itself [6]. It is quite clear that in our case, we must do supervised learning.

Neural networks are made up of artificial neurons. In the simplest terms, a neuron is a structure that can accept many inputs and can choose to fire or not to fire. Under training mode, a neuron can learn to fire or not fire when given certain inputs, and in usage mode, a neuron will correctly fire or not fire when given some input [6]. A more complicated neuron may have its inputs weighted. Many neurons are linked together to form a network, and there are two types of such networks: feed-forward and feedback. Feed-forward networks only allow data to proceed from start to finish and does not allow data to travel backwards, while feedback networks are more complex and neurons can recursively send information [6].

There are three layers to a neural network. The first layer, the one that receives the input, is the input layer which accepts the raw data fed to it. The layer that does all the computation is the hidden layer, which has that name because it is hidden from the user and is free to decide and learn on its own. The third and last layer is the output layer, which, as its name suggests, controls the output of the neural network and depends on the responses of the hidden layer [6]. When the neural network is training, it is adjusting the weights for each input, and it stops training once it realizes that it cannot improve on its current model [2]. Each time the network is trained it will produce a different output because the training data is selected at random.

3. Algorithms

We used the following code to parse out the testing data and result data so we can run each result separately with the testing data to generate the most accurate prediction model

```
allData = xlsread('trainingtest2.xlsx', 1);
values = allData(1:end, 1:3594);
Ca = allData(1:end, 3595);
P = allData(1:end, 3596);
pH = allData(1:end, 3597);
SOC = allData(1:end, 3598);
sand = allData(1:end, 3599);
realData = xlsread('realdata.xlsx');
realData2 = transpose(realData);
```

where `allData` contains a matrix of the entire spreadsheet, `values` contains only the testing data, and `Ca`, `P`, `pH`, `SOC`, and `sand` contain the results for the corresponding results. `realData` contains the data that we must model, and we must get the transpose of the matrix so that we can model it properly.

3.1. Ensemble Fit.

Ensemble learning is a form of machine learning that builds a mathematical model that correlates data matrix X and a result matrix Y. The MATLAB `fitensemble` function allows for built in ensemble learning, and in addition to the data and result matrices, it also takes in parameters about the number of learners and the number of computational cycles to run for. There are many other parameters and options that can be selected, but they are outside of the scope of this project [2]. In an attempt to find methods that would work for this data, we initially thought that the ensemble fit would give us a good starting point to refine our algorithm.

3.2. PCA.

In n-dimensional space, if there are more dimensions than points in this space, then there exists an infinite number of hyperplanes that go through these points. For example, in 2 dimensions, if we are given 1 point, there are an infinite number of lines that go through that point. Even if the number of points is $> n$, if n is very large, finding a regression model in n dimensions is computationally hard and may not be accurate, as there may be a considerable amount of random noise.

In our project, we were given 3979 dimensions to work with, and 1157 points. Although there are 3979 variables, many of them are highly correlated with each other. We can use a dimensionality reduction technique in order to reduce the number of dimensions to a number easier to work with (in our case 9), but still retain

most ($> 99\%$) of the variance (information) explained by these variables/dimensions. Essentially, we weed out all the useless parts of each dimension and compact them.

Principal Component Analysis (PCA) is a method of dimensionality reduction. Given a set of points in n dimensional space, from the results of linear algebra, we can find another basis for this space- essentially creating another orthonormal set of axis such that most of the points are clustered near just a few of these new axis. This allows us to delete most of the axis in the new set. In our case, we deleted all but 9 of the new axis. A description of how to do PCA can be found here.[2]

3.3. **Levenberg-Marquardt.** Given a set of n points in k-dimensional space, we want to find a model that fits those n points such that the difference of the square differences is minimum, that is:

$$\sum_{i=1}^{n} |Y_i - Y_i'|^2$$

where each Y_i is a point we are given and each Y_i' is the point in our model.

Levenberg-Marquardt finds a model that minimizes this by using a gradient descent method; the details of this algorithm can be found in Marquardt's original paper published in 1963.[3]

The basic idea of Levenberg-Marquardt is analogous to the following example in 3 dimensions:

> Suppose we are hiking at a park and we want to find the lowest point in the park. We start off at a random location. A gradient descent algorithm tells us to look at all directions, calculate the steepest direction, and go in that direction for some distance based on how steep it is. A bit of randomness is added to this because we might be end up in the middle of a bowl shaped area with mountains on all sides, even though the true lowest point is outside those mountains. Levenberg-Marquardt is a type of gradient descent algorithm that calculates the linear derivative to determine which direction to go in, but as this derivative decreases (meaning we sense we are close to the minimum), it calculates the quadratic derivative to determine which direction to go in for a more accurate result. Like any gradient descent algorithm, Levenberg-Marquardt works best when the function we are trying to minimize is convex, or if the entire park is visible from any point we are standing. This algorithm works well for machine learning because most of the regression functions are close to convex.

In our project, we use the Levenberg-Marquardt algorithm to find the minimum of a 9-dimensional function. We stop when we either have 6 iterations with no improvement or the difference between 2 iterations is $< 10^{-5}$.

4. Results

To first test our submission capabilities and to provide a baseline to compare ourselves, we generated a submission with random values and submitted it, getting

[2] http://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

[3] http://epubs.siam.org/doi/pdf/10.1137/0111030

a score of 1.2 (the lower the score, better as the error is less). We knew that any submission that resulted in a number greater than this meant that our algorithm was worse than random, something that fortunately never happened. The competition also provided us with a baseline model using Bayesian Additive Regression Trees (BART), which yielded a score of 0.5.

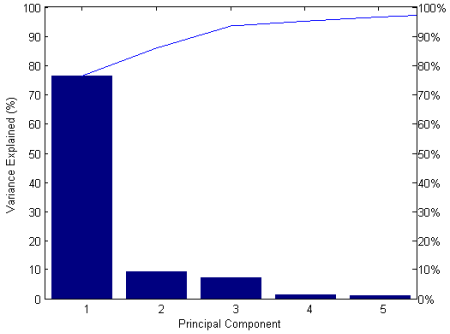
4.1. **Using Ensemble fit.** When we did ensemble fit, we ran the function with around 100 learning for 1000, 10000, and 100000 iterations, which ended up taking 2-3 hours on an Intel i7 processor. However, this brute force approach was not very accurate.

4.2. **Using Machine Learning Algorithms.** Given that there were 1157 samples and 3579 dimensions (characteristics of each sample), a simple fitting algorithm would not accurately predict the 5 target values. We tried using the Levenberg-Marquardt, Bayesian Regularization and Scaled Conjecture Gradient algorithms to perform a regression on this dataset but it took too much memory and crashed our computers often; it also did not output good results.

All but one of the dimensions were continuous variables- the last one was discrete, indicating if the sample was from topsoil or subsoil, which we translated to a discrete variable taking on a value 1 or -1. The first alternative was to split our data in half and analyze each half separately; there were 581 topsoil and 576 subsoil samples. This did not output better results than not splitting the samples, so we proceeded with keeping the data together.

4.3. **Levenberg-Marquardt With PCA.** In 3579 dimensional space, there are an infinite number of hyperplanes satisfying the 1157 points. Thus, it is reasonable to perform dimensionality reduction on the 3579 variables since there are probably high correlations between variables. We used principal component analysis (PCA) to create 3759 new variables, each a linear combination of the previous 3759 variables with the coefficients represented in a matrix. However, in this new set of variables, the first 5 represent the 96.38% of the data (Fig. 1), while the first 9 variables represent 99.1% of the data. Thus, instead of using 3759 variables in our machine learning algorithms, we used 9, which greatly reduced the space required on the computer. The Levenberg-Marquardt algorithm always performed better than the Bayesian Regularization and Scaled Conjecture Gradient algorithms. After running these algorithms multiple times for robustness (since the 1157 samples are randomly randomly permuted in the input), of the 5 target variables, we obtained good regressions for `Ca`, `pH`, `SOC`, and `sand` (Fig. 2-5) but a poor regression for `P` (Fig. 6).

FIGURE 1. (below) A pareto chart showing 5 variables explaining $> 95\%$ of the variance.



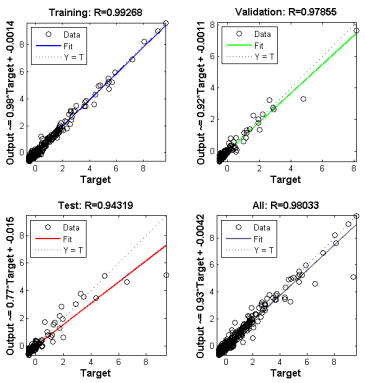


FIGURE 2. (above)

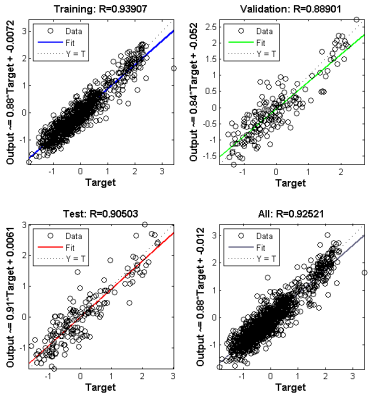


FIGURE 3. (above)

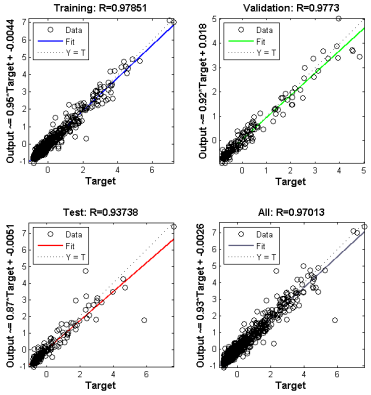


FIGURE 4. (above)

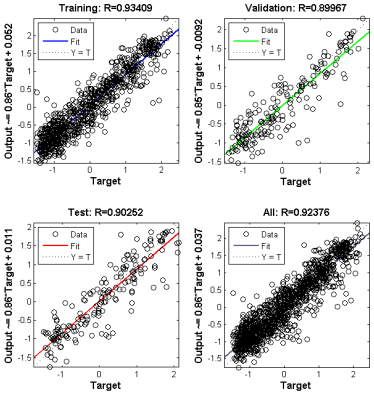
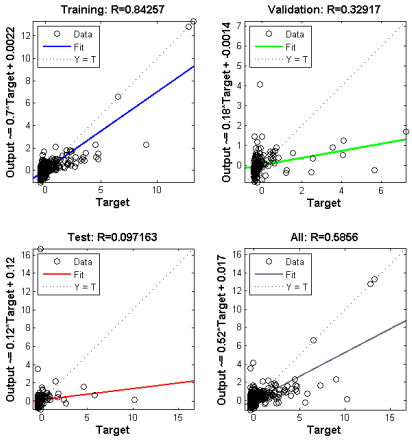


FIGURE 5. (above)

FIGURE 6. (below) Note the data clustered near $(0, 0)$ causing the model and thus the fit line to perform poorly.



To fix this, we performed an inverse hyperbolic sin (*asinh*) transformation on each of the target `P` values in the training dataset before running the Levenberg-Marquardt algorithm to obtain a regression model (Fig. 7). Similarly, we tried a log transformation; we added 0.5 to each `P` value so none of them would be negative before taking the log. The log transformation performed better (Fig. 8) although the regression was still not as good as the one for the other 4 target variables. The motivation behind this transformation is that the target variable `P` may not behave linearly. Compared to the original R value of 0.5856 for the regression for `P`, the log transformation gave a regression with R value 0.73505.

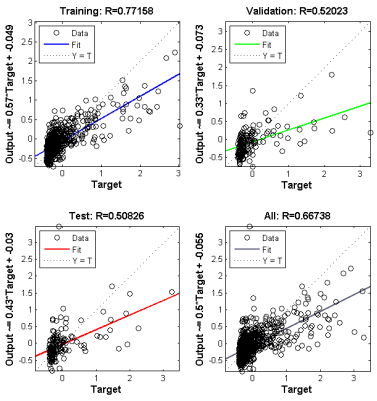


FIGURE 7. (above)

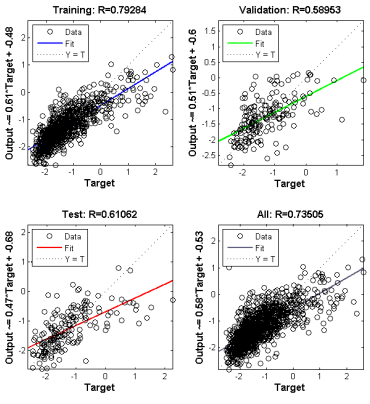


FIGURE 8. (above) Compare to Fig. 6.

In the test dataset, we performed *log* on each of the `P` values before running the model we obtained for `P`; we applied the inverse afterwards to obtain the correct

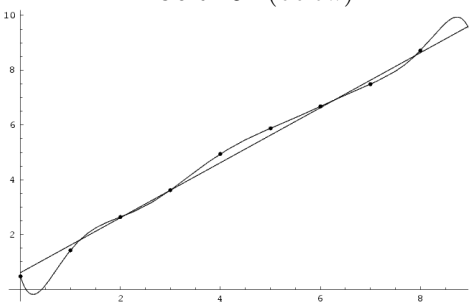
results ($2^x - 0.5$). For `Ca`, `pH`, `SOC`, and `sand`, we ran their corresponding models from the output of the Levenberg-Marquardt algorithm in the test dataset.

There were 3563 variables from infrared scans and 16 others; we tried running PCA on the 3563 variables from infrared scans, taking the first 9 variables (that explained 99% of the data) and combining that with the 16 other variables, then running Levenberg-Marquardt, but those results were worse.

4.4. **Overfitting.** One of the challenges of using an algorithm that produces a non-linear model (as we did with Levenberg-Marquardt) is overfitting the results. Overfitting is when the model tries to go through every point in the training data but ends up with a poor shape not likely to predict future target values well. For example, in Figure 9 below[4], the graph displays 2 ways to fit the points: a linear fit and a non-linear fit. The non-linear fit tries to go through all the points; the linear fit is more consistent and probably better for predicting target values.

In our case, the number of iterations were relative small (around 20) so we did not find a need to use any explicit methods to prevent overfitting, although Levenberg-Marquardt is smart about not overfitting by neglecting small errors between the model and the points.

FIGURE 9. (below)



5. CONCLUSIONS

The end result was a model with a score of 0.45143, which beat the BART benchmark. We ended up finishing 657/1233. Although we did not finish in the top 3, we were competing with many industry professionals and we learned a lot about machine learning. The 2^{nd} place finisher also used Levenberg-Marquardt with a neural network, but trained the model with 5-fold cross validation averaged over 20 times in order to prevent overfitting [5], which was, in retrospect what we should have done.

Most people would say that technology has caused climate change and degraded the environment, but as the Africa Soil Information Service shows, if used correctly technology can improve human life and help the environment. We believe that mathematics and computer science can be used to find underlying hidden patterns in environmental problems and help create solutions; we hope to do this in our future work.

[4] `http://upload.wikimedia.org/wikipedia/commons/5/5d/Overfit.png`

[5] `https://github.com/CharlyBi/Soil-Prediction`

References

1. Africa Soil Information Service, cited 2014: Key Goals of the AfSIS Project [Available online at `http://africasoils.net/about/messages`]
2. *Documentation*, MATLAB
3. Eswaran, Hari, Russell Almaraz, Paul Reich, and Pandi Zdruli, *Natural Resources Conservation Service*, Soil Quality and Soil Productivity in Africa. USDA
4. Ghodsi, Ali, *Dimensionality Reduction A Short Tutorial*, Department of Statistics and Actuarial Science University of Waterloo (2006)
5. Madsen, K., H.B. Nielsen, and O. Tingleff, *METHODS FOR NON-LINEAR LEAST SQUARES PROBLEMS.* Informatics and Mathematical Modelling Technical University of Denmark (2004)
6. Stergiou, Christos, and Dimitrios Siganos, *Neural Networks*, Neural Networks

Department of Mathematics undergrad, Brown University, Providence, Rhode Island

E-mail address: `eric_xiao@brown.edu`

Department of Computer Science undergrad, Brown University, Providence, Rhode Island

E-mail address: `martin_zhu@brown.edu`