Winning Solution for AFSIS 2014 Competition

Yasser Tabandeh

October 2014

Summary

This document describes winning solution for African Soil Property Challenge Problem (AFSIS) hosted by Kaggle. Several models such as Multilayer Perceptron, Support Vector Machines, Gaussian Process, and Multivariate Regression were combined to produce a stable framework to overcome overfitting. Each model used a different set of transformed features with different setup parameters.

The Problem Definition

The problem was to predict some soil properties (Ca, P, pH, SOC, Sand) based on the Near Infrared (NIR) data. Spectral features and spatial features were present for analysis and training. There were 1158 instances in train data and 728 instances in test data with 3578 spectral features and 16 spatial features.

Evaluation Metric

MCRMSE (mean columnwise root mean squared error) was used for evaluating quality of results. It is defined as:

$$MCRMSE = \frac{1}{5}\sum_{j=1}^{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_{ij} - a_{ij})^2}$$

Where a and p are the actual and predicted values, respectively.

Preprocessing

Different types of preprocessing were done to transform features into more relevant forms. Some of them reduce dimensionality of data and some others reduce noises.

1- **Savitzky-Golay filter**: this filter is used for smoothing the data
2- **Continuum Removal:** for normalization and handling outliers
3- **Discrete wavelet transforms:** for discrete sampling and data reduction
4- **First Derivatives:** in some cases increases prediction quality
5- **Unsupervised Feature Selection:** standard deviation was used to select top features for some algorithms.
6- **Log transform:** "P" target was transformed into log(P+1)

Modeling algorithms

1- **Neural Networks**: two types of neural network algorithms were used for training:

- Simple layer neural network (nnet package in R)
- Monotonic Multilayer Perceptron (monmlp package in R)

2- **Support Vector Machines(SVM):** svm function in e1071 package in R was used for SVM training

3- **Multivariate Regression:** mvr function in pls package in R was used for multivariate regression

4- **Gaussian Process:** gausspr function in kernlab package in R was used for Gaussian Process

For each target different algorithms were used for training. Table 1 shows detailed information for preprocessing and modeling.

Target	Model Name	Model Weight	Preprocessing Steps	Regression Algorithm
Ca	Ca_SVM1	0.100	✓ Savitzky-Golay filter	SVM, cost=1000
	Ca_SVM2	0.030	✓ None	SVM, cost=10000
	Ca_SVM3	0.100	✓ None	SVM, cost=5000
	Ca_SVM4	0.010	✓ STD Feature selection(2000 features) ✓ First Derivatives ✓ Haar dwt(3 iterations)	SVM, cost=10000
	Ca_MLP1	0.100	✓ STD Feature selection(2000 features) ✓ Haar dwt(4 iterations)	Ensemble of 10 MONMLP models (150 iterations,4 neurons in first layer and 4 neurons in second layer)
	Ca_MLP2	0.100	✓ Savitzky-Golay filter ✓ Haar dwt(4 iterations)	MONMLP (150 iterations,4 neurons in first layer and 4 neurons in second layer)
	Ca_MLP3	0.150	✓ Haar dwt(5 iterations)	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	Ca_MLP4	0.050	✓ First Derivatives ✓ Haar dwt(7 iterations)	MONMLP (150 iterations,3 neurons in first layer and 20 neurons in second layer)
	Ca_MLP5	0.030	✓ Haar dwt(4 iterations) ✓ First Derivatives	Two different MONMLP models based on Depth variable (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	Ca_MLP6	0.030	✓ Haar dwt(5 iterations)	Two different MONMLP models based on Depth variable (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	Ca_MLP7	0.150	✓ Haar dwt(4 iterations)	MONMLP (150 iterations,5 neurons in first layer and 5 neurons in second layer)
	Ca_MLP8	0.050	✓ Haar dwt(3 iterations)	MONMLP (150 iterations,5 neurons in first layer and 5 neurons in second layer)
	Ca_Gauss1	0.015	✓ None	GaussPr (rbf kernel)
	Ca_Gauss2	0.045	✓ Multiple Scatter Correction(2 iterations) ✓ First Derivatives ✓ Haar dwt(9 iterations) ✓ Partial PCA	GaussPr (rbf kernel)
	Ca_Gauss3	0.010	✓ None	GaussPr (poly kernel)
	Ca_MVR1	0.015	✓ STD Feature selection(2000 features)	MVR(120 components)
	Ca_MVR2	0.005	✓ None	MVR(100 components)
	Ca_NNET1	0.010	✓ Haar dwt(5 iterations)	NNET(10 neurons,100 iterations)
P	P_SVM1	0.088	✓ Continuum Removal	SVM, cost=5000
	P_SVM2	0.088	✓ None	SVM, cost=5000
	P_SVM3	0.088	✓ Haar dwt(1 iteration)	Two different SVM models based on Depth variable, cost=1000
	P_MLP1	0.088	✓ Savitzky-Golay filter ✓ STD Feature selection(3000 features) ✓ Haar dwt(4 iterations)	MONMLP (150 iterations,5 neurons in first layer and 0 neurons in second layer)
	P_MLP3	0.125	✓ Haar dwt(4 iterations) ✓ First Derivatives	Two different MONMLP models based on Depth variable (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	P_MLP4	0.063	✓ Haar dwt(5 iterations)	MONMLP (50 iterations,5 neurons in first layer and 5 neurons in second layer)
	P_MLP5	0.063	✓ First Derivatives ✓ Haar dwt(2 iterations) ✓ STD Feature selection(450 features)	MONMLP (50 iterations,5 neurons in first layer and 5 neurons in second layer)
	P_MLP6	0.063	✓ STD Feature selection(2500 features)	MONMLP (100 iterations,5 neurons in first layer

			✓ Haar dwt(4 iterations) ✓ First Derivatives	and 5 neurons in second layer)
	P_MLP7	0.250	✓ STD Feature selection(2500 features) ✓ Haar dwt(4 iterations) ✓ First Derivatives	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	P_MVR1	0.088	✓ Haar dwt(4 iterations)	MVR(200 components)
pH	pH_SVM1	0.116	✓ None	Two different SVM models based on Depth variable, cost=1000
	pH_SVM2	0.116	✓ None	SVM, cost=5000
	pH_MLP1	0.163	✓ Haar dwt(5 iterations)	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	pH_MLP2	0.163	✓ Haar dwt(4 iterations)	Two different MONMLP models based on Depth variable (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	pH_MLP3	0.116	✓ Haar dwt(4 iterations)	MONMLP (150 iterations,5 neurons in first layer and 5 neurons in second layer)
	pH_MLP4	0.163	✓ STD Feature selection(2500 features) ✓ Haar dwt(4 iterations) ✓ First Derivatives	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	pH_MLP5	0.163	✓ STD Feature selection(2500 features) ✓ Haar dwt(4 iterations) ✓ First Derivatives	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
SOC	SOC_SVM1	0.200	✓ None	SVM, cost=10000
	SOC_SVM2	0.140	✓ None	SVM, cost=5000
	SOC_MLP1	0.100	✓ Savitzky-Golay filter ✓ STD Feature selection(2500 features) ✓ Haar dwt(3 iterations)	MONMLP (150 iterations,3 neurons in first layer and 3 neurons in second layer)
	SOC_MLP2	0.100	✓ Savitzky-Golay filter ✓ Haar dwt(3 iterations)	MONMLP (150 iterations,3 neurons in first layer and 3 neurons in second layer)
	SOC_MLP3	0.100	✓ Savitzky-Golay filter ✓ STD Feature selection(2500 features) ✓ Haar dwt(4 iterations)	MONMLP (150 iterations,4 neurons in first layer and 4 neurons in second layer)
	SOC_MLP4	0.200	✓ First Derivatives ✓ Haar dwt(6 iterations)	MONMLP (100 iterations,4 neurons in first layer and 0 neurons in second layer)
	SOC_MLP5	0.120	✓ Haar dwt(6 iterations)	MONMLP (50 iterations,5 neurons in first layer and 5 neurons in second layer)
	SOC_MLP6	0.040	✓ STD Feature selection(2500 features) ✓ Haar dwt(4 iterations) ✓ First Derivatives	MONMLP (100 iterations,5 neurons in first layer and 5 neurons in second layer)
Sand	Sand_SVM1	0.127	✓ Savitzky-Golay filter	SVM, cost=5000
	Sand_SVM2	0.038	✓ None	SVM, cost=10000
	Sand_SVM3	0.063	✓ STD Feature selection(2000 features) ✓ First Derivatives ✓ Haar dwt(3 iterations)	SVM, cost=10000
	Sand_SVM4	0.063	✓ STD Feature selection(1500 features) ✓ First Derivatives ✓ Haar dwt(3 iterations)	SVM, cost=10000
	Sand_MLP1	0.127	✓ Haar dwt(4 iterations) ✓ Savitzky-Golay filter	MONMLP (150 iterations,4 neurons in first layer and 4 neurons in second layer)
	Sand_MLP2	0.127	✓ Haar dwt(4 iterations) ✓ PCA	MONMLP (200 iterations,5 neurons in first layer and 5 neurons in second layer)
	Sand_MLP3	0.013	✓ Haar dwt(5 iterations)	MONMLP (50 iterations,4 neurons in first layer and 0 neurons in second layer)
	Sand_MLP4	0.089	✓ Haar dwt(5 iterations)	MONMLP (50 iterations,5 neurons in first layer and 5 neurons in second layer)
	Sand_MLP5	0.152	✓ First Derivatives ✓ Haar dwt(2 iterations) ✓ STD Feature selection(450 features)	MONMLP (50 iterations,5 neurons in first layer and 5 neurons in second layer)
	Sand_MLP6	0.038	✓ Haar dwt(4 iterations) ✓ First Derivatives	Two different MONMLP models based on Depth variable (100 iterations,5 neurons in first layer and 5 neurons in second layer)
	Sand_MLP7	0.076	✓ Haar dwt(4 iterations)	MONMLP (150 iterations,5 neurons in first layer and 5 neurons in second layer)
	Sand_Gauss1	0.019	✓ None	GaussPr (rbf kernel)
	Sand_Gauss2	0.013	✓ Multiple Scatter Correction(2 iterations) ✓ First Derivatives ✓ Haar dwt(9 iterations) ✓ Partial PCA	GaussPr (rbf kernel)
	Sand_Gauss3	0.025	✓ None	GaussPr (poly kernel)
	Sand_MVR1	0.019	✓ STD Feature selection(2000 features)	MVR(120 components)
	Sand_NNET1	0.013	✓ Haar dwt(5 iterations)	NNET(10 neurons,100 iterations)

Table 1. Overall framework for modeling and training

Final predictions for each target were calculated using weighted averages of models as detailed in Table 1. Number of training rows was small in compare to number of features, so overfitting could occur.
For handling overfitting risk, the value of C parameter in SVM was set to a large number to increase regularization. Also combining different models significantly reduced drawbacks of single models. R 3.1.0 was used for overall process. This method won the competition with MCRMSE score of 0.46892.

References

Official Competition Website: http://www.kaggle.com/c/afsis-soil-properties
R package for Discrete Wavelet Transforms: http://cran.r-project.org/web/packages/wavelets/index.html
R package for processing of NIR data: http://cran.r-project.org/web/packages/prospectr/index.html
R Package for training SVM: http://cran.r-project.org/web/packages/e1071/index.html
R package for Multivariate Regression: http://cran.r-project.org/web/packages/pls/index.html
R package for Gaussian Process: http://cran.r-project.org/web/packages/kernlab/index.html
R package for Multilayer-Perceptron: http://cran.r-project.org/web/packages/monmlp/index.html
R package for Single Layer Neural Network: http://cran.r-project.org/web/packages/nnet/index.html
R package for Weka filters: http://cran.r-project.org/web/packages/RWeka/index.html