Utilization of Classification and Clustering on Analyzing Image Data

Abstract

This project focuses on classification of land cover types on objects (patches) generated from high-resolution remote sensing imagery. When segmenting the image into objects, the parameter "scale" is important – larger scale value results in bigger objects. Data are generated under 7 different scale parameters (20-140 at an interval of 20). The dataset includes 147 predictor variables (21 for each scale level). We applied PCA for dimension reduction, then conducted clustering and classification analysis by applying various techniques including K-Means, QDA, SVM, and Neural Network. To see how dimension reduction can affect the results, and what the optimal scale value for classification is, we also compared PCA-processed dataset with the original data at each scale level. Using PCA seemed to produce better performance with some but not all classification methods, and for **soil** especially, the raw data at scale 40 produced lower error rates. We found that the **soil** class actually seemed to be the most sensitive to the classification and clustering algorithms, but Neural Network, SVM and QDA were able to predict land cover class features better than the other methods. Multinomial regression performed well with the raw data, particularly at scale 40.

1. Introduction

Land cover mapping and classification has been an important application of remote sensing. It is also critical for many studies in other disciplines like geography and environmental science. Remotely sensed data are suitable for land cover mapping and classification because of its capability of observing large scale areas. Traditional land cover classification has been focusing on per-pixel approaches, assigning each pixel of the images to a particular class based only on the spectral information contained in the pixel. However, as high-spatial-resolution data are becoming more readily available, pixel-based methods have some disadvantages when applied to high-resolution data. While higher resolution allows us to observe more spatial details, such information is not utilized by pixel-based methods. Also, higher resolution will bring more noise when the pixel size gets smaller accordingly (Myint et al. 2011). Thus, object-based classification should be more powerful when used on high-resolution imagery. Object-based approaches use image segmentation algorithms to obtain "objects" consisting of multiple pixels with similar spectral information. With objects generated, we can derive shape and texture information for each object and use them with spectral information simultaneously (Rodriguez-Galiano et al. 2012).

When doing image segmentation, several parameters will affect the segment results. In this project, we are particularly interested in the parameter "scale". Smaller scale will result in smaller segments (Johnson and Xie 2013). This dataset includes variables gained at seven different scale levels (20–140 at an interval of 20). In this project, we want to see what scale level is optimal for land cover classification. Also, considering that different scale parameters may be appropriate for different land cover types (e.g. buildings are usually smaller than open grassland/forest), we will also use dimension reduction method on all variables from 7 scale levels, to see if the "synthesized" dataset can yield better results than data from a single scale level.

2. Data Description

This data set, which is a high resolution aerial image data, is obtained from UCI Machine Learning Repository and has been separated into training (168 observations) and test (507 observations) data. The study area for this data set was an urban area in Deerfield Beach, FL, USA, and contained many kinds of land cover, which was segmented into tree, grass, building, concrete, asphalt, vehicles, pools, soil and shadow by using Multiresolution Segmentation algorithm, which uses one-pixel image segments to merge neighboring segments until a heterogeneity threshold is reached.

There are mainly 21 variables with seven different scale parameters (20-140 at an interval of 20) used to analyze each segment. Scale parameters smaller than 20 make segments over-segmented, and parameters larger than 140 make segments under-segmented. Figure 1 shows the comparison of segments produced by different scales.

Table 1 is the list of details of these 21 variables and 1 response variable. To be clearer, the variables showed in the Table 1 (except for the response variable) were measured repeatedly for seven times, so there are total 147 variables in this data set.

Table 1 : List of features

Data Type	Attribute	Description
Response Variable (nominal)	Class	Land Cover Class
Shape Variable	BrdIndex	Border Index
	Round	Roundness

	Compact	Compactness
	ShpIndx	Shape Index
	LW	Length/Width
	Rect	Rectangularity
	Dens	Density
	Assym	Asymmetry
	BordLngth	Border Length
Size Variable	Area	Area in m2
Spectral Variable	Bright	Brightness
	Mean_G	Green
	Mean_R	Red
	Mean_NIR	Near Infrared
	NDVI	Normalized Difference Vegetation Index
Texture Variable	SD_G	Standard Deviation of Green
	SD_R	Standard Deviation of Red
	SD_NIR	Standard Deviation of Near Infrared
	GLCM (1, 2 and 3)	Gray-Level Co-occurrence Matrix

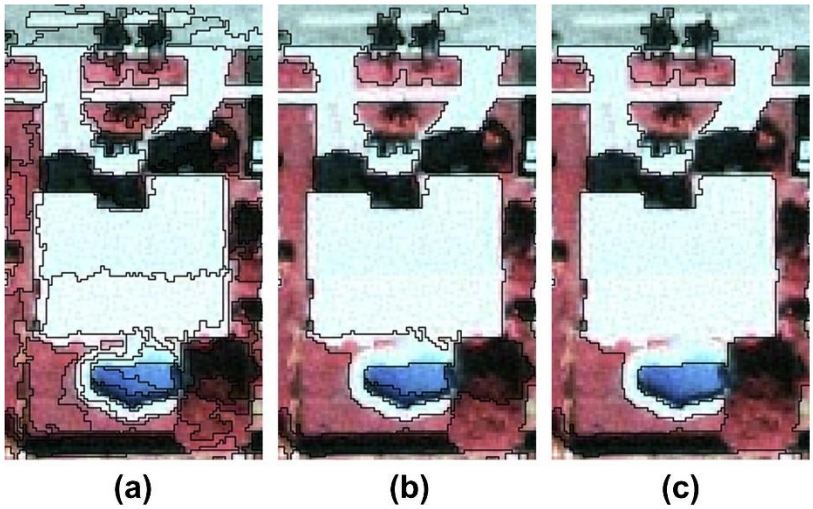


Figure 1 : (a) scale 20 (b) scale 80 and (c) scale 140 segments overlaid on the color infrared imagery for a subset of the image. [1]

3. Data Exploration

In order to understand this data more, we performed some preliminary analysis first. As mentioned before, there are many variables in this data, and some of them are very similar, therefore, we checked the correlation between these main variables. From Figure 1, we could know that there are high correlation between these similar variables. By applying PCA, we could reach significant dimensional reduction.

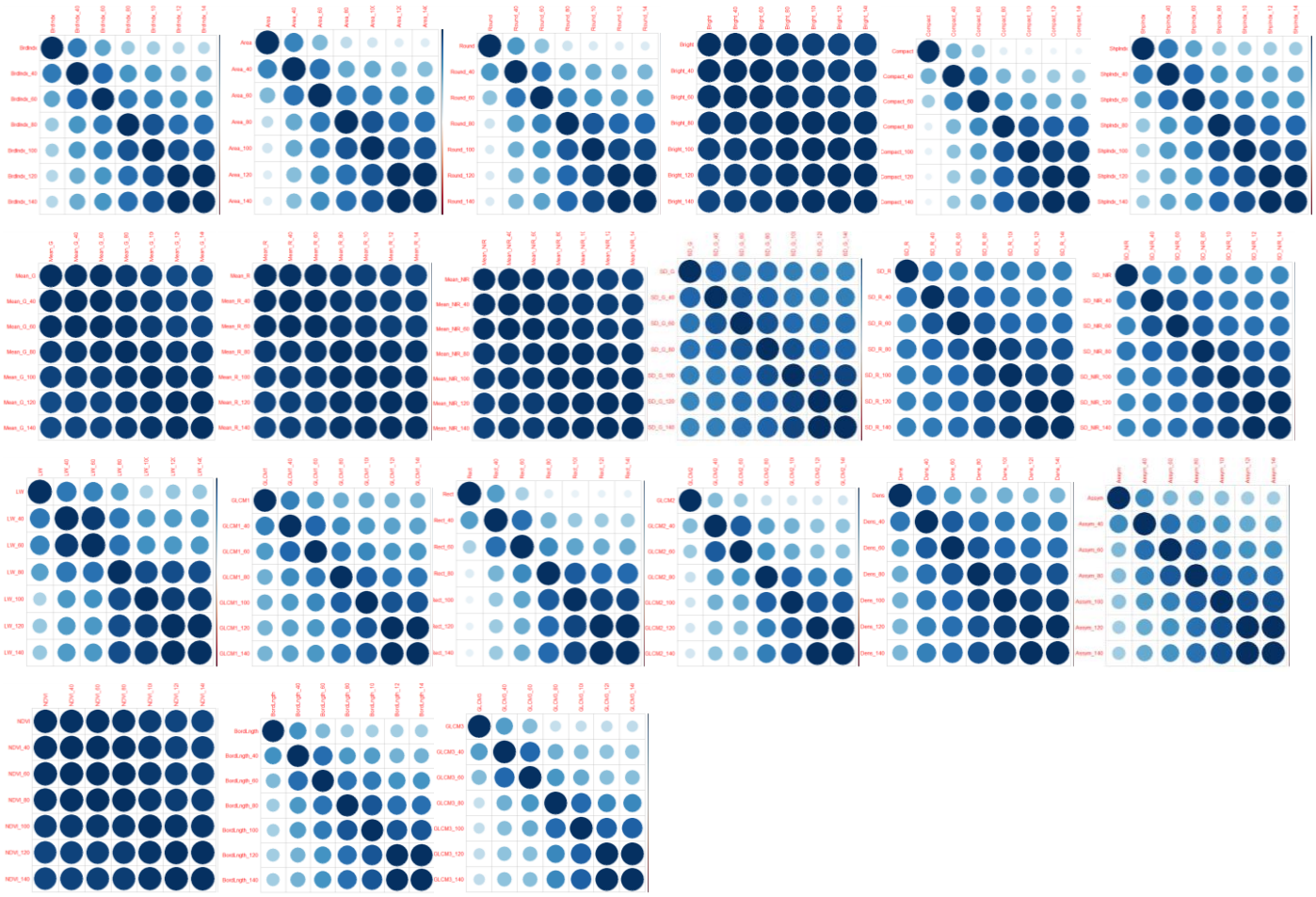


Figure 2 : Correlation plots between main variables at different scales, from left to right and top to bottom, they are BrdIndx, Area, Round, Bright, Compact, ShpIndx, Mean_G, Mean_R, Mean_NIR, SD_G, SD_R, SD_NIR, LW, GLCM1, Rect, GLCM2, Dens, Assym, NDVI, BoardLngth and GLCM3

4. Pre-process Data

Before doing any deeper analysis, we needed to solve two problems of this data. First, because these variables were measured in different scales, we decided to standardize all variables in avoid of overweighting a variable with larger range. Second, there are 147 variables in the original data, and the variables with different coarse scales but the same measurement (such as Round_40 and Round_140) are highly correlated, so we applied PCA (principal components analysis) for dimensional reduction.

By applying PCA, we not only reduced the dimension, but also made the new variables (components) independent of each other. In this data, there are (V_1, V_2 , V_3, ……, V_{147}) variables, and after performing PCA, our new variables are given by

$$P_i = \sum_{j=1}^{147} w_{i,j} V_j,$$

which maximize $w_i{}^T \Sigma\, w_i$ subject to $w_i{}^T w_i = 1, w_i{}^T w_j = 0$, and w_i is the eigenvector corresponds to the largest eigenvalue of the decomposition of Σ ($\mathrm{W} \Lambda W^T$).

From Table 2, we got the result of cumulative amount of variation explained by new components. The variation explained by one component is calculated by $\frac{\lambda_i}{\sum_{i=1}^{147} \lambda_i}$, where λ_i is the largest eigenvalues. In order to make the predication

of classification and clustering more accurate (some methods do not perform well on the high dimension), we decided to choose the first seven components as our new variables. In addition, the first seven components have explained more than 70% variation of this data. Besides, by Figure 3, the scree plot shows a bend in the elbow around at comp.7.

For the convenience of performing the result, we combined the training and test data and did the standardization and dimensional reduction together, and then we separated it into new training and test data by the ratio of 3:1.

Table 2 : Percent of variation explained

Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**	**Comp.6**	**Comp.7**	**Comp.8**	**Comp.9**	**Comp.10**	**Comp.11**	**Comp.12**	**Comp.13**	**Comp.14**
0.273	0.444	0.546	0.621	0.672	0.717	0.745	0.77	0.792	0.811	0.826	0.839	0.852	0.864

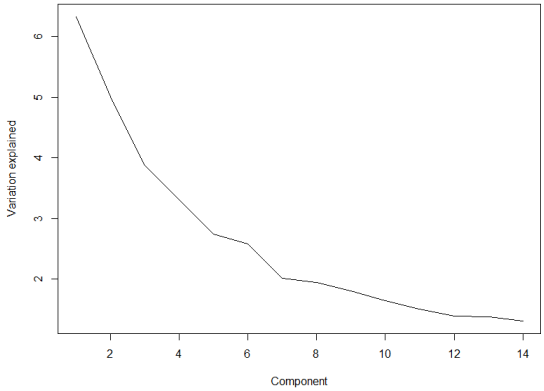


Figure 3 : Scree plot for PCA

5. Classification

5.1 QDA

In this part, we used one generative parametric method to do the classification, which is QDA (quadratic discriminant analysis). QDA and LDA (linear discriminant analysis) are very similar, both of them assume the observations from each class are drawn from Gaussian distribution, and then plug the estimated parameters into Bayes rule to compare the posterior probability and assign the observation to the class with the largest probability. However, there is one thing different, when applying LDA, we assume that mean and variance are the same for each class, but when applying QDA, we do not make constant variance assumption, which makes QDA more flexible than LDA. Besides, it is obvious that the boundary between each class is non-linear, it would be more appropriate and reasonable to apply QDA. Given data pairs (x_i,y_i) for i=1,2,……,507, where y_i is the class label for x_i, we assumed that P(X|Y = k)=p_k is Gaussian distribution with mean μ_k and variance Σ_k, where k=1,2,3……9. Therefore, we could know that

$$p_k = \frac{1}{2\pi^{p/2}|\Sigma_k|^{1/2}} \exp\left(\frac{-1}{2}(x-\mu_k)^T {\Sigma_k}^{-1}(x-\mu_k)\right).$$

Then we used the optimal classifier, which is $\text{argmax}_k \delta_k(x) = \text{argmax}_k P(Y=k|X=x)$ to decide which class should the observation be assigned to. In this classifier function, $\delta_k(x)$ is the discriminant function, which is given by

$$\delta_k(x) = \frac{-1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T {\Sigma_k}^{-1}(x-\mu_k) + log\pi_k, where\ \pi_k = P(Y=K).$$

Table 3 shows the training error rate, test error rate and cross-validation error rate of QDA predication. Training error rate and test error rate were calculated by applying QDA on the training data and do the prediction on the training data and test data, respectively. Cross-validation error rate was calculated by applying QDA on the whole data (combining

training and test data). In addition, we also calculated the error rates for data on different scale, which all have 21 features. By Table 3, we could know that no matter for which version, the test error rate is always the highest. Besides, except for training error rate, most of the test and CV error rates are a little higher when we performed QDA on data based on different scales. However, for the scale 40, the test error is lower, which is also the lowest test error rate. We focused on the scale 40, and found out the training error rate is also the lowest. Although the CV error rate is not the lowest, it is third lowest one. Therefore, for QDA, the the data with features on scale 40 performs the best.

Table 3 : Error rates of performing QDA

	Training error	**Test error**	**CV error**
PCA-preprocessed	0.134123	0.2202381	0.16459614
Scale_20	0.0375	0.238	0.207
Scale_40	0.0355	0.202	0.201
Scale_60	0.0493	0.238	0.215
Scale_80	0.069	0.232	0.234
Scale_100	0.0809	0.286	0.268
Scale_120	0.114	0.321	0.311
Scale_140	0.142	0.304	0.3305

5.2 Multinomial Regression

Unlike QDA and LDA , which are generative methods and can be complex when processing high dimensional data, multinomial regression is a discriminative model which holds no assumptions on p(x), and only needs the conditional distribution P(Y=k|X=x). In multinomial regression, we used a link function, which is logit function in this study, to model the probability of y being one of the classes.

We applied multinomial regression on the 7 sets of variables generated with 7 different scale parameters (20, 40, 60, 80, 100, 120, 140). Table 4 shows the train, test, and cv errors for the 7 sets of variables. As we see, scale 40 seems to be the best scale parameter among the 7 scale values. Table 5 is the confusion matrix for the original datasets with scale 40.

Table 4 : Classification Errors for Multinomial Regression using original data

	s20	**s40**	**s60**	**s80**	**s100**	**s120**	**S140**
train	0.110	0.108	0.118	0.143	0.161	0.205	0.226
test	0.226	0.220	0.273	0.226	0.261	0.25	0.291
cv	0.213	0.201	0.215	0.230	0.264	0.303	0.325

Table 5 shows the confusion matrices using original data and PCA-preprocessed data. Interestingly, we see here the class **soil** has much higher accuracy than using the PCA-preprocessed dataset. However, the class **building** is largely misclassified with **concrete**. Considering that soil, building and concrete actually have similar spectral response. It is possible that the scale parameter has an important effect when classifying these classes. Table 6 shows the accuracy for building and soil for each of the 7 scale parameters using multinomial regression. From table 6 we see that building's accuracy gets lower when scale gets higher, while soil's accuracy gets higher at the same time, except for scale 140. It seems that scale 140 is able to classify **building** and **soil**, however, at the scale value, **trees** and **grass** are highly misclassified with each other, so the overall accuracy is not boosted.

Table 5a : Confusion Matrix for Multinomial Regression using PCA-preprocessed data

	a	b	c	d	e	f	g	h	i
A	13	0	0	0	0	0	1	0	0
B	0	24	0	4	0	0	0	1	0
C	0	3	9	0	0	0	0	0	0
D	0	4	0	20	0	0	0	2	0
E	0	0	0	0	20	1	0	2	4
F	0	0	0	0	0	6	1	0	0
G	3	0	0	0	0	0	16	0	0
H	0	1	0	1	1	0	0	1	1
I	0	0	0	0	4	0	1	0	24

Table 5b : Confusion Matrix for Multinomial Regression using original data

	a	b	c	d	e	f	g	h	i
a	15	0	0	0	0	0	5	0	0
b	0	20	0	3	0	0	0	0	0
c	0	0	8	1	0	0	0	0	0
d	0	10	1	20	0	1	0	1	0
e	0	1	0	0	18	1	0	0	3
f	0	0	0	0	0	5	0	0	0
g	1	0	0	1	0	0	14	0	0
h	0	1	0	0	3	0	0	5	0
i	0	0	0	0	4	0	0	0	26

Note: a-asphalt, b-building, c-car, d-concrete, e-grass, f-pool, g-shadow, h-soil, i-tree. Labels on top are reference classes, and labels on left are predicted classes.

Table 6 : Accuracy for Building and Soil with Multinomial Regression using original data

	s20	s40	s60	s80	s100	s120	s140
building	0.5312500	0.6250000	0.6875000	0.71875	0.7812500	0.7500000	0.6875000
soil	0.8333333	0.8333333	0.6666667	0.50000	0.1666667	0.3333333	0.8333333

5.3 Support Vector Machine

Support Vector Machine (SVM) is a very popular and well-performing classifier. The algorithm seeks to find a hyperplane that best separates the different classes in the dataset. Compared to LDA/QDA, SVM is more able to deal with high dimensional datasets because of the use of kernels. In finding the best parameters for SVM to best classify the dataset, we started with experimenting with 3 types of kernels - linear, polynomial, and radial - with changing cost parameter. Again, we started with PCA-preprocessed data.

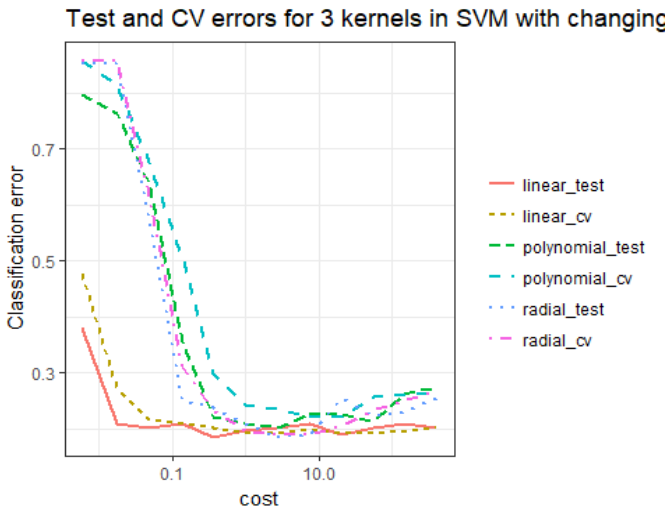


Figure 4 : Test and CV errors for 3 kernels in SVM with changing cost

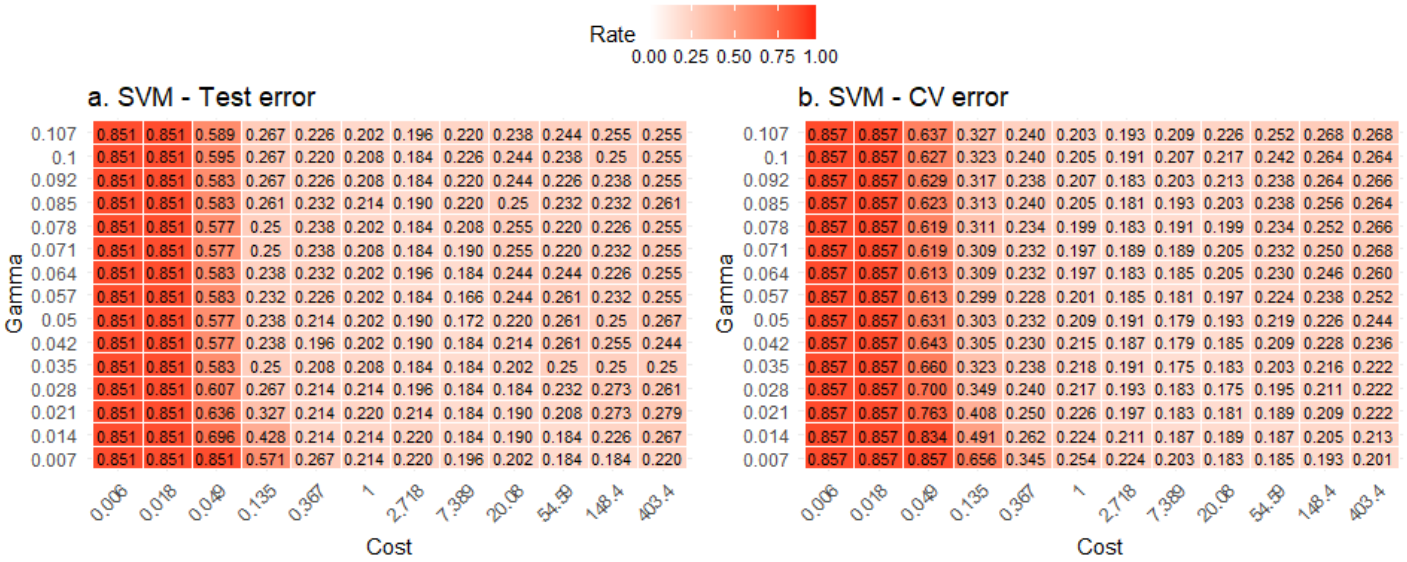


Figure 5 : Test and CV errors for radial kernel changing cost and gamma

Figure 4 shows the test and CV errors for 3 kernels in SVM with changing cost. Generally, by tuning cost, we see that polynomial kernel performs worst among the three kernels. Linear kernel and radial kernel have similar performance when only tuning cost. To find the best classification accuracy, we further tuned the gamma parameter in the radial kernel.

Table 7. Best error rates for SVM - radial kernel

train error	test error	cv error
0.0710059	0.1666667	0.1756358

Table 8 : Best error rates for SVM - linear kernel

train error	test error	cv error
0.1262327	0.1845238	0.1933411

Figure 5 shows test and CV errors for radial kernel with changing cost and gamma. From this table we can see the best overall error rates are 0.167 of test error when cost = exp(2), gamma = 0.8/7, and 0.176 of cv error when cost = exp(2), gamma = 1/7 (also showed in table 7). For comparison, table 8 shows the best error rates for linear kernel. We see that SVM with linear kernel has better performance than multinomial regression, which may imply that most data in this dataset can be separated linearly. But we do see lower error rates when using radial kernels. Radial kernels can project data to higher-dimensional space in a more complex way, which can be useful for some cases where linear separation does not perform well.

Figure 6 shows the heatmap of classification accuracy for both linear and radial kernel using PCA-preprocessed data. Again, we see some poorer performance with regard to **soil**, with radial kernel has a bit better accuracy rate for **soil**.

Again, we are interested in how SVM perform when using original data generated at 7 different scale parameters. Based on previous analysis, the radial kernel performs best, so here we tuned the cost and gamma parameters for each of the 7 sets of the original dataset. Table 9 and 10 show the best test and cv error achieved at each scale level. We see, as in multinomial regression, that scale 40 seems to be the best scale in terms of the overall test accuracy. The test error rate achieves as 0.143, which is pretty low.

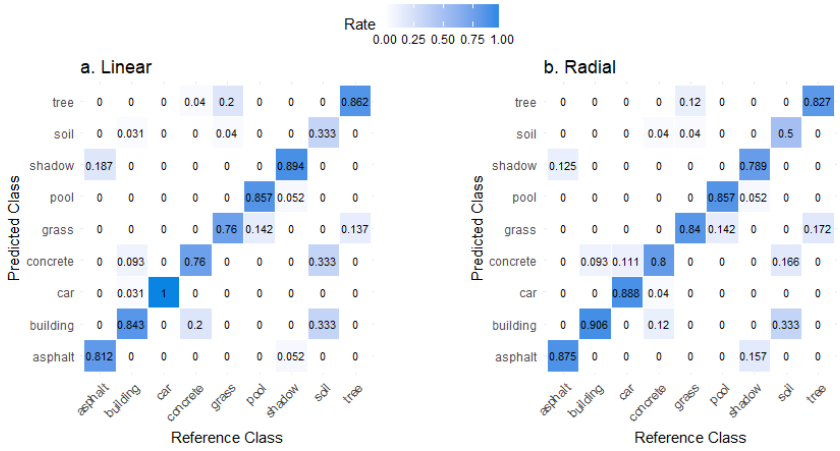


Figure 6 : Heatmap of Accuracy/Error Rates for SVM using PCA-processed data

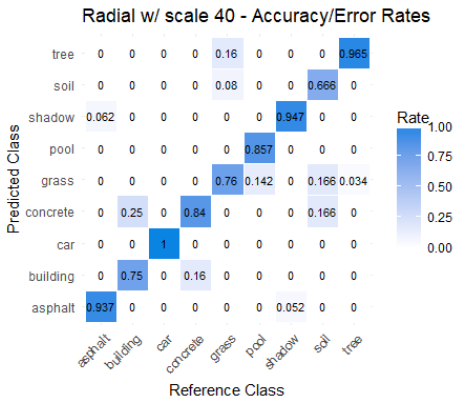


Figure 7 : Heatmap of Accuracy/Error Rates for SVM using scale 40 original data

We chose to use test error here for comparison instead of cv errors because: (1) we tried several times randomly splitting to dataset into training and testing dataset, and found that each time SVM radial kernel with scale 40 can yield similar test error rates (around 14.5%), though each time the lowest test error rates may appear at different cost/gamma values. (2) Based on (1), we believe that for each different split of train/test data, the parameters for SVM radial kernel may be a bit different to achieve optimal results. So when doing cross validation, the training data are different each time while using the same set of parameters (cost and gamma), so the cv error may not represent the best performance.

Figure 7 shows the heatmap of accuracy using SVM radial kernel with scale 40 dataset. From this figure we see that results are more accurate overall, and performance on **soil** also improved a little bit.

Table 9 : Best test error rates at various scales

	s20	**s40**	**s60**	**s80**	**s100**	**s120**	**S140**
test	0.1845238	0.1428571	0.1666667	0.172619	0.202381	0.202381	0.2380952

Table 10 : Best cv error rates at various scales

	s20	s40	s60	s80	s100	s120	S140
cv error	0.1972821	0.1933799	0.1874782	0.1952436	0.2269462	0.2681033	0.2800427

5.4 Random Forest

Random forest is one of the ensemble classifiers. It creates multiple decision trees with bootstrapped training datasets and only include a randomly selected subset of variables for each split. The output class will be decided by the most voted class from the all created trees. Because of bootstrap and random select of variables, the problem of overfitting can be controlled. Thus, random forest usually has better performance than individual decision trees. Also, random forest can assess the importance of each variable. As above, we first tried random forest on the PCA-preprocessed dataset.

Table 11 shows the train, test, and cv errors for random forest. It seems that when using PCA-processed data, the error rates are worse than most of other classifiers that we tried before (Multinomial Regression, SVM, etc.). The heatmap (Figure 8(a)) also shows that random forest performed worse especially on soil, with accuracy of 0. Since trees tend to behave better where they are able to have better performance, and perform even worse where data are hard to be classified. This result may not be very unexpected.

Since previous experiments in multinomial regression and SVM show that using original dataset may help improve accuracy of some certain classes like soil, we also tried applying random forest on original data at 7 scale levels.

Table 11 : Errors for random forest on PCA processed data

Train error	Test error	CV error
0.2209073	0.2202381	0.2228888

Table 12 : Errors for random forest on original data

	Scale_20	Scale_40	Scale_60	Scale_80	Scale_100	Scale_120	Scale_140
Training	0.1775148	0.1814596	0.2110454	0.2189349	0.2465483	0.3076923	0.3392505
Test	0.1964286	0.1369048	0.1547619	0.1726190	0.1785714	0.2083333	0.2440476
CV	0.1913803	0.2012619	0.2052805	0.2149874	0.2524558	0.2977092	0.3273539

As mentioned above, random forest is able to yield the importance of variables. Figure 8 shows the variable importance among the variables generated at scale 40.

From Figure 9, we see that the most important variable is NDVI, followed by **Mean_Red**, **Mean_NearInfraRed**, **Brightness**, and **Mean_Green.** It is obvious that the most important variables for correctly classification of land cover are these **spectral** variables. NDVI, as an index representing the abundance of vegetation, are critical for identifying trees, grass, and other types of land cover objects with little vegetation cover. Interestingly, the second and third important variables are **Mean_Red** and **Mean_NearInfraRed.** NDVI is calculated from: NDVI = (NIR - R) / (NIR + R). As vegetation is extremely reflective in NIR, vegetation reflects barely no red light. Thus, red and NIR, as NDVI, are all important spectral information for identifying vegetation. Note that **SD of Red, NIR, and Green** are also closely following, indicating spectral

information's importance for classification. After these spectral variables, we see both shape variables (**Area**, **Shape Index**, and **Border Index**) and texture variables (**GLCM1**, **GLCM2**).

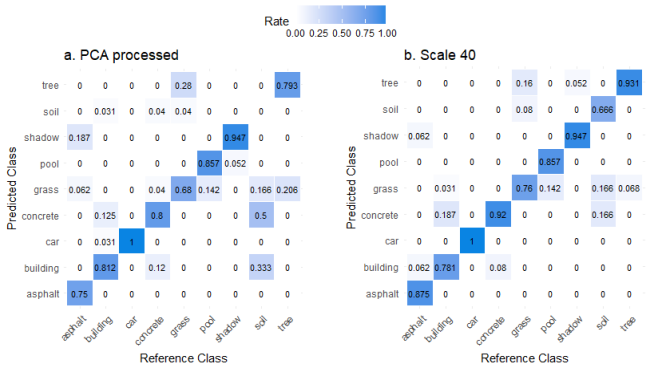


Figure 8 : Heatmap of Accuracy/Error Rates for random forest using PCA-preprocessed data and scale 40 data

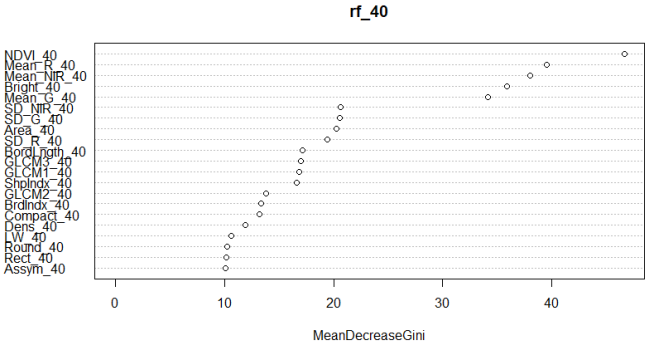


Figure 9 : Variable importance generated from random forsestusing scale 40 data

The variable importance figure may also imply why in some cases using the original dataset can be better than the PCA-processed data. As PCA captures the most of the variance, which is very likely to be contributed by spectral variables, some information from shape and texture variables may be left out. While spectral information can classify most of the classes, it is not enough for those classes with similar spectral response. For example, trees and grass may be similar in spectral response, but different in their texture (grass smoother and trees rougher). Thus, it really depends on our purpose to choose what kind of dataset to be used. If we are just interested in the vegetation information, it seems like PCA-preprocessed data are good. But if we are interested in more detailed information (like subclasses in vegetation), or some classes with similar spectral information as others, like soil, then using original dataset can be a better choice.

5.5 K-Nearest Neighbors

To estimate class densities, the K Nearest Neighbor classifier was applied on the first 7 components of the PC transformed dataset. It was also applied to the raw data at different scales to compare and determine which has the better performance in predicting the test class. To maximize prediction performance and achieve the smallest possible prediction error, we have used a 10-fold cross validation on the train set to evaluate the accuracy of k = 1, 3, 5, 10, 15, 20, 25, 30, 35, 40. The cross validation shows k = 10 has the highest accuracy (Figure 10).

After running the KNN model with k = 10, we tested the model by evaluating the test error, the train error, as well as the 10-fold cross validation error (Table 13). The train error rate is approximately 18%, the test error is approximately

22% and the CV error is approximately 23%. This error rate is fairly high and shows KNN as a classifier might not be the best for predicting landcover. We compare these results to those from running the KNN on different scales of the data. We find that on average, the PCA preprocessed data is much more accurate for prediction of land cover. The scaled raw data has train, test and CV error rates at approximately 40% and higher (Table 14). This emphasizes that for prediction, an aggregation of the aerial image data at different scales instead of any one scale is more accurate (Figure 11).

Table 13

Training error	Test error	CV error
0.1893491124	0.2142857143	0.2268244576

Table 14

	Scale_40	Scale_60	Scale_80	Scale_100	Scale_140
Training	0.3905	0.3925	0.4003	0.4497	0.503
Test	0.494	0.5417	0.5357	0.5952	0.625
CV	0.5128	0.5384	0.5247	0.5502	0.639

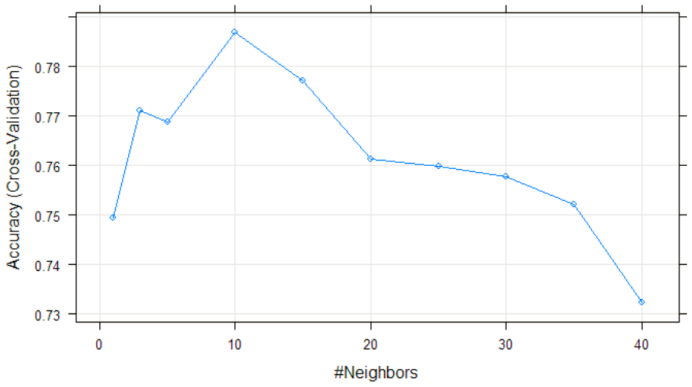


Figure 10 : 10-Fold cross validation for optimal 'k'

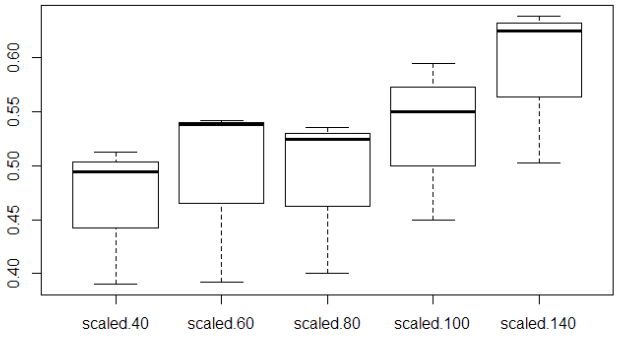


Figure 11 : Raw data error

5.6 Neural Networks

To run the neural net, we encoded the data and created dummy variables representing each class coding 1 or 0 to indicate what class a row is. For deciding the number of neurons to use in the hidden layer for our model, we reviewed the literature for the rules of thumb for choosing the optimal number of neurons without spending an large amount of computing time trying a sequence of neurons. The rules of thumb we used include; taking into account the input layer and output layer of our data and using the $N_h = \frac{N_s}{(\alpha*(N_i+N_o))}$ formula, ensuring that $N_h = 2/3N_i + N_o$ and $N_h < 2N_i$. After running a 10-fold cross validation on 6 different estimates of the number of neurons in the hidden layer, we computed the scaled CV error and RMSE. The results show that the number of neurons in the hidden layer with the lowest mean cv error rate is 8 (Table 15). Compared to the KNN classifier, the neural net is a better classifier for prediction on the test set (Figure 12). This suggests that the neural net has a fairly high sensitivity to the data.

Table 15. Prediction error

	NN 5	NN (3,2)	NN 8	NN 10	NN (10,3)	NN15
RMSE	3.2488	2.9574	3.2223	3.2986	3.4022	3.2394
CV	10.5732	8.9524	11.0613	10.8929	11.6875	10.5054

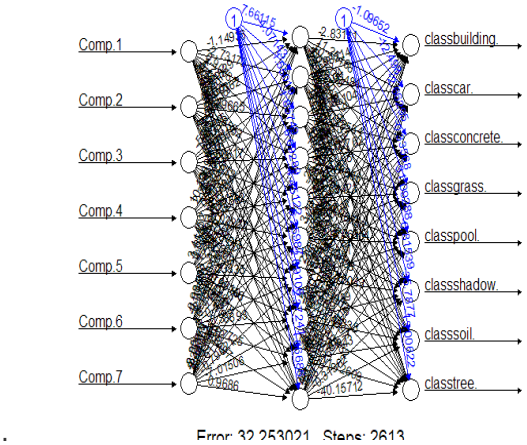


Figure 12. Neural Net Hidden Layer Neurons = 8

6. Clustering—k-means

In this section, we chose one unsupervised method to perform the clustering. There are several steps when applying k-means. First, a set of $\{\mu_k\}_{k=1}^{K}$ are initialized arbitrarily to serve as the initial means for K clusters. Second, the distance between means and each data point will be calculated by the method of Euclidean distance, which is defined by $\sum_{n=1}^{N}\sum_{k=1}^{K}\|X_n - \mu_k\|^2$. Then a data point would be assigned to the cluster with the shortest distance between it and the mean of that cluster. That is, for n=1, ……, N,

$$\text{the cluster assignment } Z_n^k := \begin{cases} 1, if\ k = argmin_{i \le K}\|X_n - \mu_k\| \\ 0, otherwise \end{cases}.$$

Third, after including a new data point, the mean of each cluster will be updated by

$$\mu_k = \frac{\sum_{n=1}^{N} Z_n^k X_n}{\sum_{n=1}^{N} Z_n^k}, \text{ for k=1, ……, K.}$$

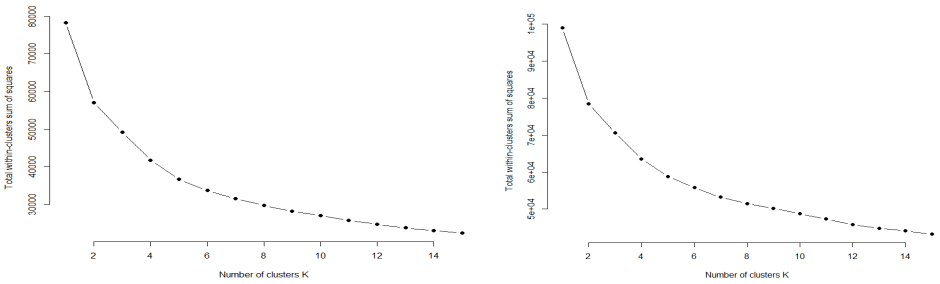
And then the whole procedure will keep repeating the second and third steps until convergence.

Figure 13 represents the variance within the clusters, which shows a decreasing trend when the number of cluster increases. From this figure, we could see a bend at k=6, which indicates that six clusters may be our optimal choice. In addition, we also compared two versions of data, one is (a) PCA-preprocessed data, the other is (b) scaled original data. The trend of plot (a) and plot (b) does not show many different. No matter which kind of data we use in the first place, we would choose six clusters. However, the values of total within-clusters sum of squares are different. It is obvious that the value of plot (b) is higher than plot (a), which suggests that PCA-preprocessed version is more desirable.

In order to compare with the original cluster, we also chose nine clusters to perform k-means and do the visualization. We chose nine clusters for two plots of Figure 14 only because of the convenience of visualization including original classification. By comparing two plots of Figure 14, we know that the PCA-preprocessed version is better, since most of the data points with the same shape (original class) have the same color (new cluster). In addition, the data points in the same new cluster seem to be closer to each other on the plot (a).

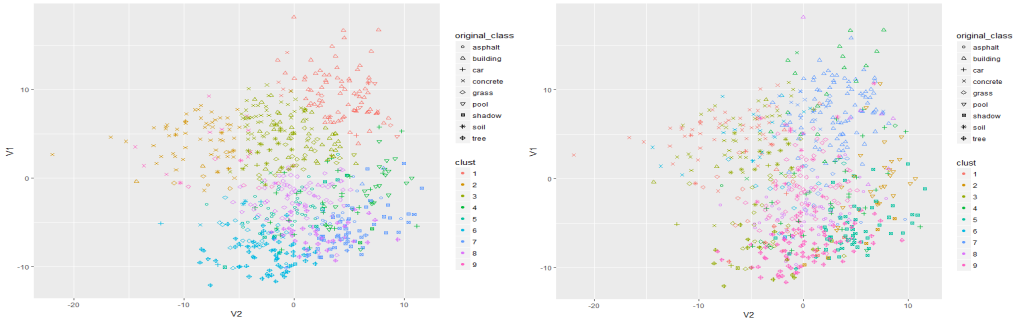
We also generated Silhouette plot to estimate the wellness of this clustering. The Silhouette plot measures how close each data point in one cluster is to other points in the neighboring clusters, and the value of coefficients is between (1, -1). On Figure 15, we chose six clusters to compare the Silhouette plot for PCA-preprocessed data and original data. From Figure 15, we observed that the silhouette value is larger on plot (a), which indicates that the distance of each data point

to other points in the neighboring cluster is larger when we used PCA-preprocessed data. In addition, the number of negative value, which means this data point might be assigned to the wrong cluster, is also less when applying on PCA-preprocessed data. Therefore, for k-means, PCA-preprocessing data does improve the quality of clustering.



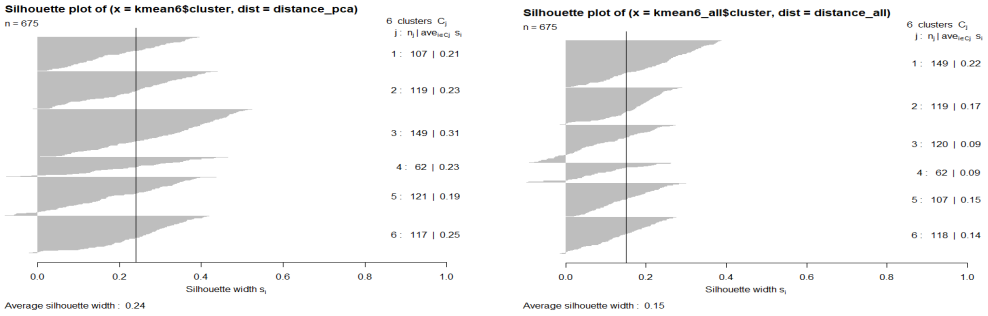
(a) (b)

Figure 13 : Total within-clusters sum of squares of (a) pca-preprocessed and (b) original data with respect to number of clusters



(a) (b)

Figure 14 : Scatter plots in 2-dim of (a) pca-preprocessed and (b) original data after performing k-means



(a) (b)

Figure 15 : Silhouette plots for (a) pca-preprocessed data and (b) original data

7. Conclusion

After performing classification and clustering methods, both parametric and non-parametric on the first 7 principal components as well as subsets of the raw data at different scales, we found the **soil** class to be the most sensitive to classification and clustering methods.

With PCA-preprocessed data, Neural Network and SVM seems to be the best performing in terms of test error and was able to predict land cover features using our data with a CV error of around 8.9%, significantly lower than the next best performing classifier, SVM which had a test error rate of 16.7%. QDA performs similarly in the CV error rate, with a

16.4% error, but the test error rate was fairly higher at 22%. The random forest and KNN classifiers performed similarly to the QDA with a test error rate of approximately 22% and 21% respectively.

In terms of raw data, we did see some low test error rates when running QDA and multinomial regression on the scale data, particularly with classifying **soil** using the scale 40 parameter. We also found that SVM performed well using each scale parameter subset of the raw data. QDA and Random Forest also performed well especially when using scale 40 subset of the raw data. However, KNN performed the worst with substantially high error rates across all scale subsets.

By comparing QDA and multinomial regression, we noticed that the last one performed slightly better when applying on raw data, which is not surprising, since multinomial regression has less assumption and works better on high dimensional data. For SVM, we noticed the importance of choosing kernel, and the result of radial kernel is better than linear kernel suggests that the boundary between each class might not be linear. However, the radial kernel is also more complicated, so if the error rate does not increase too much, in some certain cases, we might wat to choose linear kernel. For the random forest, it performed better when using raw data. Since this method is designed to classify high dimensional data, this result is not surprising. For KNN, the performance of PCA-preprocessed data is better, which might indicate the curse of dimensionality. Since the distance of KNN is mostly calculated by Euclidean distance, which tends to be meaningless in high dimension.

As for clustering method—k-means, the performance seems to be better when using the PCA-preprocessed data, since the average of silhouette value is higher, which also indicate that the method might suffer from the curse of dimensionality. Since in most case (including our case), the definition of distance of k-means is Euclidean distance, which always becomes meaningless in the high dimension.

PCA processing of the data didn't necessarily yield better performance for clustering data points and land cover features and we found a fairly high number of data points were assigned to the wrong cluster. Perhaps because PCA captured most of the variance contributed by spectral information, but left out some texture and shape information which may be useful for some identifying some classes like soil (as discussed in Random Forest section). Classification of land cover features has been an important part of environmental sustainability efforts, and in our analysis we were able to achieve a CV error rate less than 10% - with the Neural Network.

8. Reference

[1] Johnson, B., Xie, Z., 2013. Classifying a high resolution image of an urban area using super-object information. ISPRS Journal of Photogrammetry and Remote Sensing, 83, 40-49.

[2] Johnson, B., 2013. High resolution urban land cover classification using a competitive multi-scale object-based approach. Remote Sensing Letters, 4 (2), 131-140.

[3] UCI Machine Learning https://archive.ics.uci.edu/ml/datasets/Urban+Land+Cover

[4] Myint, S. W., Gober, P., Brazel, A., Grossman-Clarke, S., & Weng, Q. (2011). Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery. Remote sensing of environment, 115(5), 1145-1161.

[5] Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. ISPRS Journal of Photogrammetry and Remote Sensing, 67, 93-104.