Chapter 5: Classification

Chris Holden

03/24/2015

Introduction

In this chapter we will classify the Landsat image we've been working with using a supervised classification approach which incorporates the training data we worked with in chapter 4. Specifically, we will be using the RandomForest (Brieman 2001) ensemble decision tree algorithm by Leo Breiman and Adele Cutler. The RandomForest algorithm has recently become extremely popular in the field of remote sensing, and is quite fast when compared to some other machine learning approaches (e.g., SVM can be quite computationally intensive). This isn't to say that it is the best per se; rather it is a great first step into the world of machine learning for classification and regression.

A few good resources for understanding RandomForest can be found:

- Breiman, Leo. 2001. Random Forests. Machine Learning 45-1: 5-32.
- Wikipedia - Random Forest
- Breiman and Cutler's website
- Blog entry from yhat

A brief explanation of the RandomForest algorithm comes from the name. Rather than utilize the predictions of a single decision tree, the algorithm will take the ensemble result of a large number of decision trees (a forest of them). The "Random" part of the name comes from the term "bootstrap aggregating", or "bagging". What this means is that each tree within the forest only gets to train on some subset of the full training dataset (the subset is determined by sampling with replacement). The elements of the training data for each tree that are left unseen are held "out-of-bag" for estimation of accuracy. Randomness also helps decide which feature input variables are seen at each node in each decision tree. Once all individual trees are fit to the random subset of the training data, using a random set of feature variable at each node, the ensemble of them all is used to give the final prediction.

In the classification mode, this means that if you were to have 5 classes being predicted using 500 trees, the output prediction would be the class that has the most number of the 500 trees predicting it. The proportion of the number of trees that voted for the winning class can be a diagnostic of the representativeness of your training data relative to the rest of the image. Taking the 500 trees example, if you have pixels which are voted to be in the "Forest" land cover class by 475 of 500 trees, you could say that this was a relatively certain prediction. On the other hand, if you have a pixel which gets 250 votes for "Forest" and 225 votes for "Shrub", you could interpret this as either an innately confusing pixel (maybe it is a mixed pixel, or it is a small statured forest) or as an indicator that you need more training data samples in these types of pixels.

Finally, Random Forest has some other benefits: + It gives you a measure of "variable important" which relates how useful your input features (e.g. spectral bands) were in the classification + The "out-of-bag" samples in each tree can be used to validate each tree. Grouping these predicted accuracies across all trees can sometimes give you an unbiased estimate of the error rate similar to doing cross-validation. + Can be used for regressions, unsupervised clustering, or supervised classification + Available in many popular languages, including Python, R, and MATLAB + Free and open source, and fast

RandomForest in R

RandomForest is available in the `randomForest` package.

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

RandomForest can be used for regression, supervised classification, and unsupervised classification, or clustering. Since we already have categorical training data, we'll be doing supervised classification to predict land cover labels.

Data

Just in case it's not in your workspace, let's download and load the data. In this final chapter we'll be using a cloud-free image also available on Github:

```
library(raster)
```

```
## Loading required package: sp
```

```
library(rgdal)
```

```
## rgdal: version: 0.9-2, (SVN revision 526)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 1.11.2, released 2015/02/10
## Path to GDAL shared files: /usr/share/gdal/1.11
## Loaded PROJ.4 runtime: Rel. 4.8.0, 6 March 2012, [PJ_VERSION: 480]
## Path to PROJ.4 shared files: (autodetected)
```

```
if (file.exists('training_data.shp') == F) {
    download.file(url = 'https://raw.githubusercontent.com/ceholden/open-geo-tutorial/master/example/tra
                  destfile = 'training_data.zip', method = 'curl')
    unzip('training_data.zip')
}
training <- readOGR('training_data.shp', layer='training_data')
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "training_data.shp", layer: "training_data"
## with 30 features
## It has 2 fields
```

```
if (file.exists('LE70220491999322EDC01_stack.gtif') == F) {
    download.file(url='https://raw.githubusercontent.com/ceholden/open-geo-tutorial/master/example/LE70
                  destfile='LE70220491999322EDC01_stack.gtif', method='curl')
}
le7 <- brick('LE70220491999322EDC01_stack.gtif')
```

With the data loaded, let's extract the reflectance data covering the training data labels.

```
roi_data <- extract(le7, training, df=TRUE)
```

Remember that this dataset may have clouds or cloud shadows. Let's mask them out:

```
roi_data[which(roi_data$Band.8 > 1)] <- NA
```

We'll also need to attach the labels to this `DataFrame`:

```
roi_data$lc <- as.factor(training$id[roi_data$ID])
roi_data$desc <- as.factor(training$class[roi_data$ID])
```

Train RandomForest

Let's use the extracted training data to train the RandomForest algorithm:

```
# Set seed value for RNG for reproducibility
set.seed(1234567890)

colnames(roi_data)
```

```
##  [1] "ID"                  "band.1.reflectance" "band.2.reflectance"
##  [4] "band.3.reflectance" "band.4.reflectance" "band.5.reflectance"
##  [7] "band.7.reflectance" "band.6.temperature" "Band.8"
## [10] "lc"                  "desc"
```

```
# Shorten column names
colnames(roi_data) <- c('ID', 'b1', 'b2', 'b3', 'b4', 'b5', 'b7', 'b6', 'fmask', 'lc', 'desc')

rf <- randomForest(lc ~ b1 + b2 + b3 + b4 + b5 + b7 + b6, data=roi_data, importance=TRUE)
print(rf)
```

```
##
## Call:
##  randomForest(formula = lc ~ b1 + b2 + b3 + b4 + b5 + b7 + b6,      data = roi_data, importance = TR
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1.67%
## Confusion matrix:
##     1  2   3   4  5 class.error
## 1 382  0   1   0  0 0.002610966
## 2   0 16   0   0  0 0.000000000
## 3   1  0 144   0  0 0.006896552
## 4   0  0   0 102  4 0.037735849
## 5   0  0   0   6 62 0.088235294
```

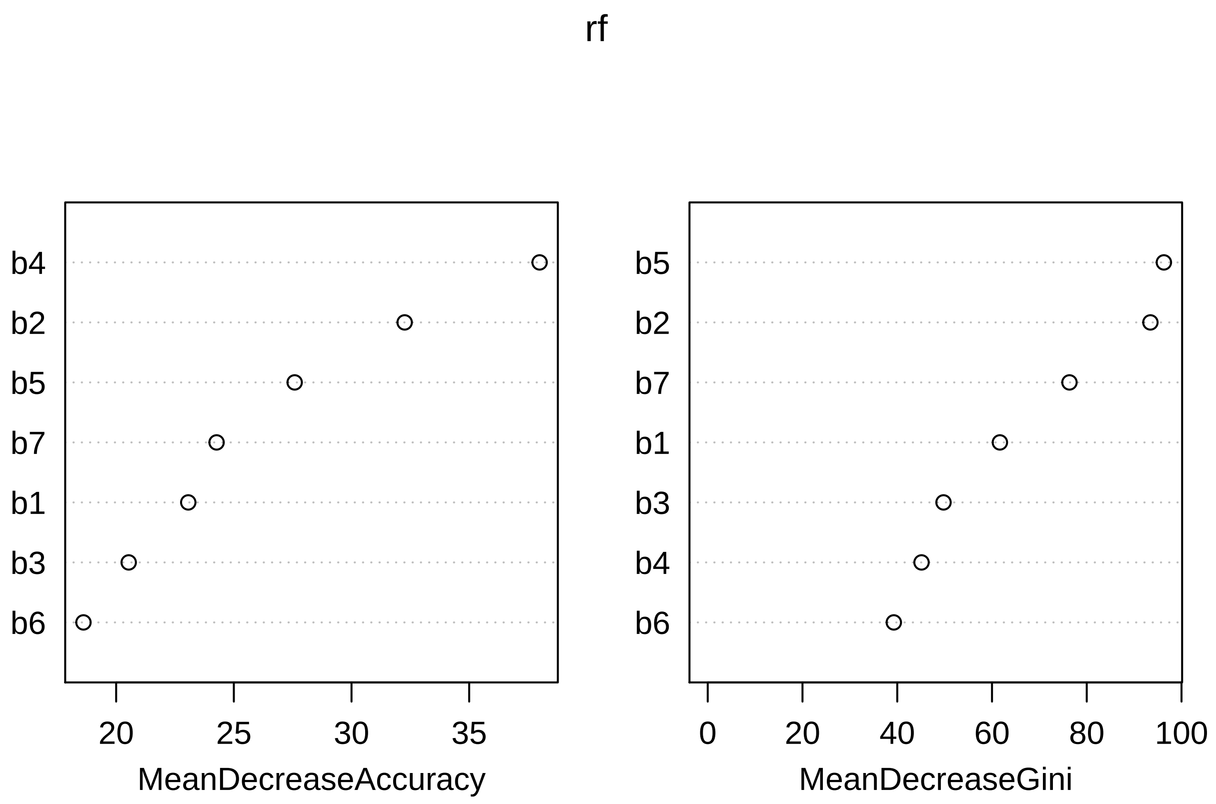
One of the useful metrics provided by RandomForest is the `importance` metric. This metric can be computed in two ways – the mean decrease in accuracy as each variable is removed or the Gini impurity metric. Either way it gives an idea of what features (Landsat bands) were important in assigning the classification labels.

See `?importance` for more information:

Here are the definitions of the variable importance measures. The first measure is computed from permuting OOB data: For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification, MSE for regression). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences. If the standard deviation of the differences is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case).

The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares.

```
varImpPlot(rf)
```



Sticking with similarities for right now, we see that the thermal band (band 6) is not very useful for our classification. This should agree with our expectations because the brightness temperature does not vary greatly across the land covers we selected, although it can prove useful when comparing irregated agriculture against natural grasslands or urban heat island impacted development against more suburban development.

I am more inclined to believe the importance as measured by the mean decrease in the Gini impurity metric because it ranks the two short-wave infrared (SWIR) bands more highly than the near-infrared (NIR) or the visible. Because it is impacted by leaf moisture, the SWIR bands greatly help the distinction between forests and grasslands. One thing to note is that because many of the bands are intrinsically or physically correlated with one another, our dataset is highly collinear. This multicollinearity means that a small subset of the bands provide truely unique information and our importance metric will be unstable depending on the random chance of one band being favored over another highly correlated band.

Classify

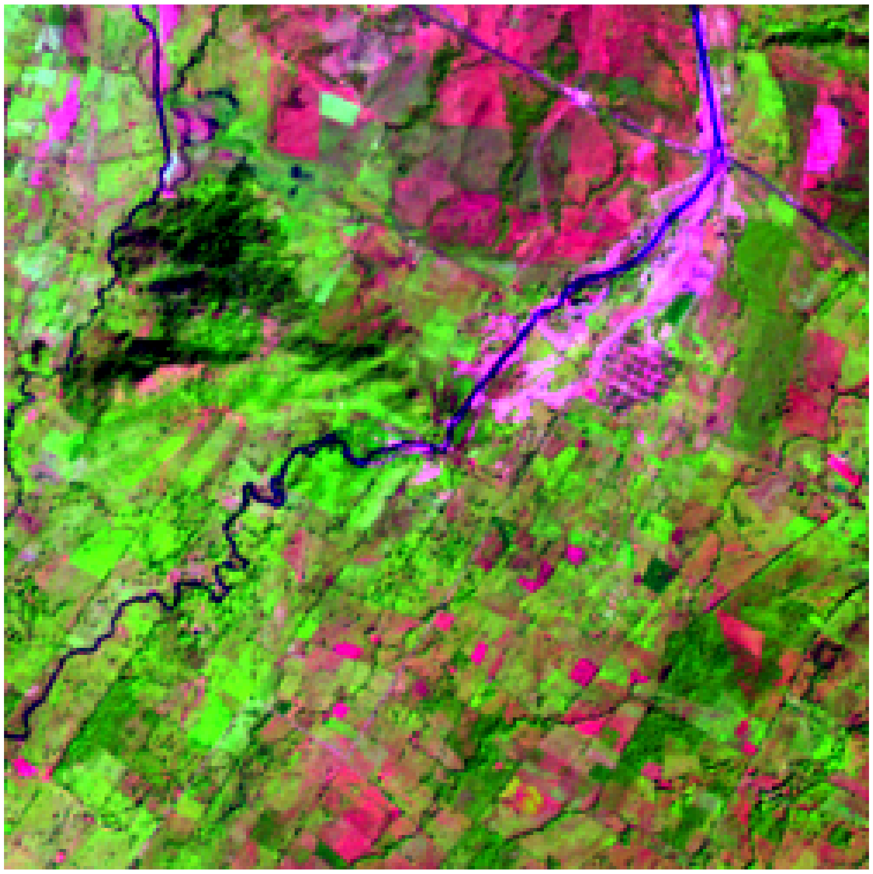
We can proceed to classify the entire raster now that our algorithm is trained. To make sure our classifier knows what raster bands to choose from, we'll begin by renaming the layers within our Landsat 7 image.

```r
le7_class <- le7
names(le7_class) <- c('b1', 'b2', 'b3', 'b4', 'b5', 'b7', 'b6', 'fmask')

# Predict!
le7_pred <- predict(le7_class, model=rf, na.rm=T)
```

Map

Let's make a map!

```r
# Create color map
colors <- c(rgb(0, 150, 0, maxColorValue=255),  # Forest
            rgb(0, 0, 255, maxColorValue=255),  # Water
            rgb(0, 255, 0, maxColorValue=255),  # Herbaceous
            rgb(160, 82, 45, maxColorValue=255),  # Barren
            rgb(255, 0, 0, maxColorValue=255))  # Urban

plotRGB(le7, r=5, g=4, b=3, stretch="lin")
```



```r
plot(le7_pred, col=colors)
```