Section V: Random Forests and GAMs

450C

Stanford University

Department of Political Science

Toby Nowacki

Zuhad Hai

Overview

Random Forests: Intuition

- We want to compute the average $\bar{y}$ for every partition of the data, where the partition is a unique combination of covariates.
- Why is the curse of dimensionality a problem here?

Random Forests: Intuition

- Random Forests give us a way out by searching for the best way to split the multidimensional space
- Within each region, compute the average value of y
- But how to find optimal region?
- Greedy algorithm: tries to find partition that satisfies local minimum of prediction error
- What can go wrong with the greedy algorithm?

Random Forests: Intuition

- To mitigate concern, we introduce random sampling across variables (select z of the J variables)
- When different variables are selected, we will also observe different nodes / trees!
- In general, no good advice on how deep we should grow these trees / how many trees we want
- Tree depth comes at a bias-variance tradeoff: the less data we have in each node, the more do we run the risk of overfitting.
- Can do crossvalidation!

Random Forests: Implementation

Let's prepare our data.

```r
library(randomForest)
library(mlbench)
library(caret)

data(Sonar)
df <- Sonar
x <- df[, 1:50]
y <- df[, 51]
```

Random Forests: Implementation

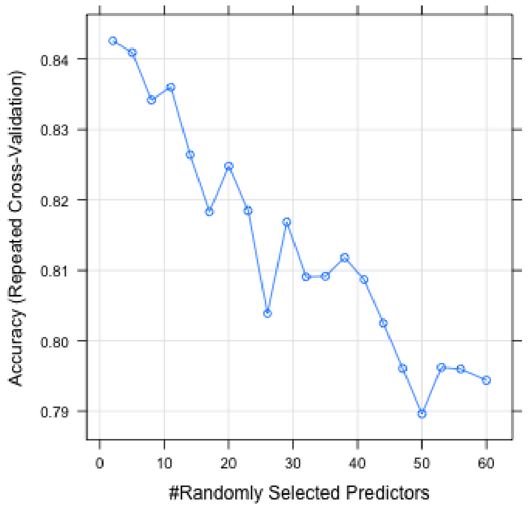
Fit the model.

```
set.seed(2020)
control ← trainControl(method = "repeatedcv",
                        number = 10, repeats = 3)
metric ← "Accuracy"
rf_random ← train(Class ~ ., data = df, method = "rf",
  metric = metric, tuneLength = 20, trControl = control)
```

Random Forests: Implementation

Accuracy by tree length:

```
plot(rf_random)
```



Random Forests: Implementation

```r
tg ← expand.grid(.mtry = c(10:20))
rf_grid ← train(Class ~ ., data = df, method = "rf",
  metric = metric, tuneGrid = tg, trControl = control)
```

Random Forests: Implementation

```
print(rf_grid)
```

```
## Random Forest
##
## 208 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 188, 188, 187, 187, 187, 187, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   10    0.8366089  0.6685702
##   11    0.8368543  0.6695420
##   12    0.8415296  0.6787108
##   13    0.8286003  0.6521362
##   14    0.8287518  0.6527369
##   15    0.8223160  0.6390713
##   16    0.8207359  0.6358603
##   17    0.8173232  0.6290108
##   18    0.8140765  0.6217789
##   19    0.8124820  0.6193741
```

Generalised Additive Models: Intuition

- GAMs introduce non-linearity into our classic regression framework:

$$y_i = \beta_0 + s_1(x_{1i}) + s_2(x_{2i}) + s_3(x_{3i}) + u_i$$

where the functions s_1 etc. are estimated from the data.

- Theory somewhat involved, but the key takeaway is that we rely on partial residuals (the relationship between x_1 and y after controlling for the rest)
- GAMs allow us to interpret the relationship between any variable and the outcome in a bivariate plot
- Crucial to remember that the plots show changes in y *relative to its mean*
- Interactions can be modelled with GAMs, but quickly run into the curse of dimensionality problem again.

Generalised Additive Models: Implementation

- Let's compare OLS and GAM results.
- Data and example taken from Peisakhin and Rozenas (2018)
- How does exposure to Russian propaganda media sources affect political behaviour?

```r
d ← read.csv("data.csv")
d ← na.omit(d)

head(d)
```

```
##   precinct  oblast places noplaces type size ukrainian district14par
## 1   590884 Сумська   СУМИ        1 city    3     77.44          157
## 2   590885 Сумська   СУМИ        1 city    3     77.44          157
## 3   590886 Сумська   СУМИ        1 city    3     77.44          157
## 4   590887 Сумська   СУМИ        1 city    3     77.44          157
## 5   590888 Сумська   СУМИ        1 city    3     77.44          157
## 6   590889 Сумська   СУМИ        1 city    3     77.44          157
##   registered14par voted14parl oppblock14par porosh14par  r14parl turnout14
## 1            1552         844    0.04976303   0.2500000 13.98104     0.543
## 2            2368        1370    0.04087591   0.2503650 11.24088     0.578
## 3            1564         887    0.03720406   0.2559188 11.49944     0.567
## 4            2152        1252    0.05191693   0.2739617 11.50160     0.581
```

Generalised Additive Models: Implementation

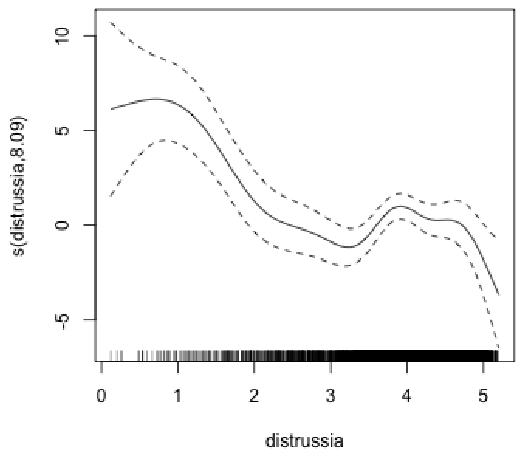
```r
form0 <- formula("r14pres ~ qualityq + distrussia + factor(Raion) + u

m0 <- lm(form0, data = d)
coeftest(m0, vcovCL(m0, cluster = m0$model[["factor(Raion)"]]))
```

```
##
## t test of coefficients:
##
##                               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                   4.619296   5.363981  0.8612 0.3892039
## qualityq                      6.431126   2.616363  2.4580 0.0140180 *
## distrussia                   -2.012976   0.725696 -2.7739 0.0055690 **
## factor(Raion)Balakliis       20.586593   3.044345  6.7622 1.586e-11 ***
## factor(Raion)Barvinkivs      13.964029   2.717876  5.1378 2.931e-07 ***
## factor(Raion)Bilopils        -7.047792   1.514835 -4.6525 3.401e-06 ***
## factor(Raion)Blyzniukivs      7.403772   2.365735  3.1296 0.0017650 **
## factor(Raion)Bobrovyts       -0.717404   0.311357 -2.3041 0.0212743 *
## factor(Raion)Bohodukhivs      8.789799   2.669802  3.2923 0.0010036 **
## factor(Raion)Borivs           8.143906   2.761783  2.9488 0.0032114 **
## factor(Raion)Borznians       -0.310447   0.328731 -0.9444 0.3450404
## factor(Raion)Buryns          -4.047601   0.919786 -4.4006 1.112e-05 ***
## factor(Raion)Chernihivs      -2.687138   0.646097 -4.1590 3.273e-05 ***
## factor(Raion)Chuhu‹vs        20.010818   2.723104  7.3485 2.483e-13 ***
## factor(Raion)Derhachivs      16.403024   2.501040  6.5585 6.237e-11 ***
```

Generalised Additive Models: Implementation

```
form1 <- formula("r14pres ~ qualityq + s(distrussia) + factor(Raion)
m1 <- gam(form1, data = d)
plot(m1)
```



Generalised Additive Models: Implementation

```
form2 <- formula("r14pres ~ qualityq + s(distrussia) + s(ukrainian) -

m2 <- gam(form2, data = d)
plot(m2)
```

Midterm revision

- What have we covered so far?
  - Maximum Likelihood
  - Probit and Logit: Estimation and Uncertainty
  - Principal Components Analysis
  - Ridge, LASSO and Naive Bayes
  - Random Forests, Ensemble Methods and GAMs
- You should be comfortable with:
  - fitting these models to data
  - interpreting the model output
  - evaluating the model's fit, strengths and weaknesses
  - critically applying these techniques to new problems
- We do **not** expect you to:
  - solve complex algebra or other mathematical problems
  - develop new code for applications outside of class