Landcover Classification Model

Owen Xu Li

April 2024

Landcover Classification Model

```
load('data/labeled_points.Rdata')

labeled = labeled %>%
  select(ID, landcover)

dl <- labeled_train %>%
      left_join(labeled, by = 'ID') %>%
      mutate(veg = ifelse(landcover %in% c('natforest', 'orchard', 'cropland'), 1, 0),
             NDVI100 = NDVI*100,
             NDBI100 = NDBI*100)
```

Let's compute mean band values for each location.

```
dm <- dl %>%
      group_by(ID) %>%
      summarise(
        B1   = mean(B1, na.rm=T),
        B2   = mean(B2, na.rm=T),
        B3   = mean(B3, na.rm=T),
        B4   = mean(B4, na.rm=T),
        B5   = mean(B5, na.rm=T),
        B6_VCID_1 = mean(B6_VCID_1, na.rm=T),
        B6_VCID_2 = mean(B6_VCID_2, na.rm=T),
        B7        = mean(B7, na.rm=T),
        NDVI100   = mean(NDVI100, na.rm=T),
        # NDBI100   = mean(NDBI100, na.rm=T), ## causes warnings with lasso
        # EVI  = mean(EVI, na.rm=T),          ## not relevant
        landcover = unique(landcover),
        veg = unique(veg)) %>%
      select(-ID, -landcover)


ht(dm)
```

```
## # A tibble: 2 x 10
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>
## 1  80.4  69.2  73.4  91.4  79        130.      148.  51.3    15.1     1
## 2  83.5  73.5  78.9  94.6  87.1      131.      150   57.9    13.4     1
##   ...
## # A tibble: 2 x 10
```

```
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>
## 1  88.8  79.3  89.7  98.3  104.      133.      153.  70.1    8.41     1
## 2  92.2  87   102.  101.   118.      134.      155.  88.5    2.06     1
```

We are interested in modeling the probability of `veg` as a function of the mean band values. We will compare the following four models:

- logistic regression (with no regularization) using only `NDVI100`
- logistic regression (with no regularization) using all band values
- logistic regression with Ridge regularization using all band values
- logistic regression with Lasso regularization using all band values

1. Models with all of the data

Fit models with all of the data and find (in-sample) predicted probabilities for each observation. Explore the trace curves and discuss any notable observations.

```
## logistic regression with no regularization using only NDVI100
m1 <- glm(veg ~ NDVI100, data = dm, family = binomial)
dm$predm1 <- predict(m1, type = 'response', newdata = dm)
summary(m1)
```

```
##
## Call:
## glm(formula = veg ~ NDVI100, family = binomial, data = dm)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.1318     0.2599  -0.507    0.612
## NDVI100       0.7953     0.1029   7.729 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 449.87  on 399  degrees of freedom
## Residual deviance: 105.95  on 398  degrees of freedom
## AIC: 109.95
##
## Number of Fisher Scoring iterations: 8
```

```
## logistic regression with no regularization using all band values
m2 <- glm(veg ~ ., data = dm, family = binomial)
dm$predm2 <- predict(m2, type = 'response', newdata = dm)
summary(m2)
```

```
##
## Call:
## glm(formula = veg ~ ., family = binomial, data = dm)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 178.4566   116.6565   1.530  0.12608
## B1            0.9808     0.4390   2.234  0.02549 *
## B2           -1.6828     0.6943  -2.424  0.01537 *
## B3            1.0323     0.5807   1.778  0.07547 .
## B4           -0.5766     0.5156  -1.118  0.26340
## B5            0.6270     0.2004   3.129  0.00175 **
## B6_VCID_1    -4.1567     2.5178  -1.651  0.09876 .
## B6_VCID_2     2.3382     1.4277   1.638  0.10149
## B7           -0.5541     0.2218  -2.499  0.01246 *
## NDVI100       1.7048     0.9351   1.823  0.06830 .
## predm1       -2.7813     4.1185  -0.675  0.49946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 449.868  on 399  degrees of freedom
## Residual deviance:  54.646  on 389  degrees of freedom
## AIC: 76.646
##
## Number of Fisher Scoring iterations: 10
```

```
## logistic regression with ridge regularization using all band values
x <- model.matrix(veg ~ B1 + B2 + B3 + B4 + B5 +
                    B6_VCID_1 + B6_VCID_2 + B7 + NDVI100, data = dm)[,-1]
y <- dm$veg
m3 <- cv.glmnet(x, y, family = 'binomial', alpha = 0)
dm$predm3 <- predict(m3, newx = x, s = 'lambda.1se', type = 'response')
summary(m3)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  13    lognet list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
## logistic regression with lasso regularization using all band values
m4 <- cv.glmnet(x, y, family = 'binomial', alpha = 1)
dm$predm4 <- predict(m4, newx = x, s = 'lambda.1se', type = 'response')
summary(m4)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  13    lognet list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

To find the trace curves, we need to plot the coefficients as functions of lambda:

```
coefs.m3 <- coef(m3, s = m3$lambda)
coefs.m4 <- coef(m4, s = m4$lambda)
colnames(coefs.m3) = paste0('lambda', round(m3$lambda,6))
colnames(coefs.m4) = paste0('lambda', round(m4$lambda,6))

coefs.m3   <- coefs.m3   %>%
  as.matrix() %>%
```

```
  as.data.frame() %>%
  rownames_to_column() %>%
  pivot_longer(cols=-rowname) %>%
  mutate(model='Ridge')

coefs.m4   <- coefs.m4   %>%
  as.matrix() %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  pivot_longer(cols=-rowname) %>%
  mutate(model='Lasso')

# bind rows
coefs1 <- bind_rows(coefs.m3, coefs.m4) %>%
  mutate(name = as.numeric(gsub('lambda', '', name))) %>%
  filter(rowname!='(Intercept)') %>%
  rename(lambda=name,
         var = rowname)

head(coefs1)
```

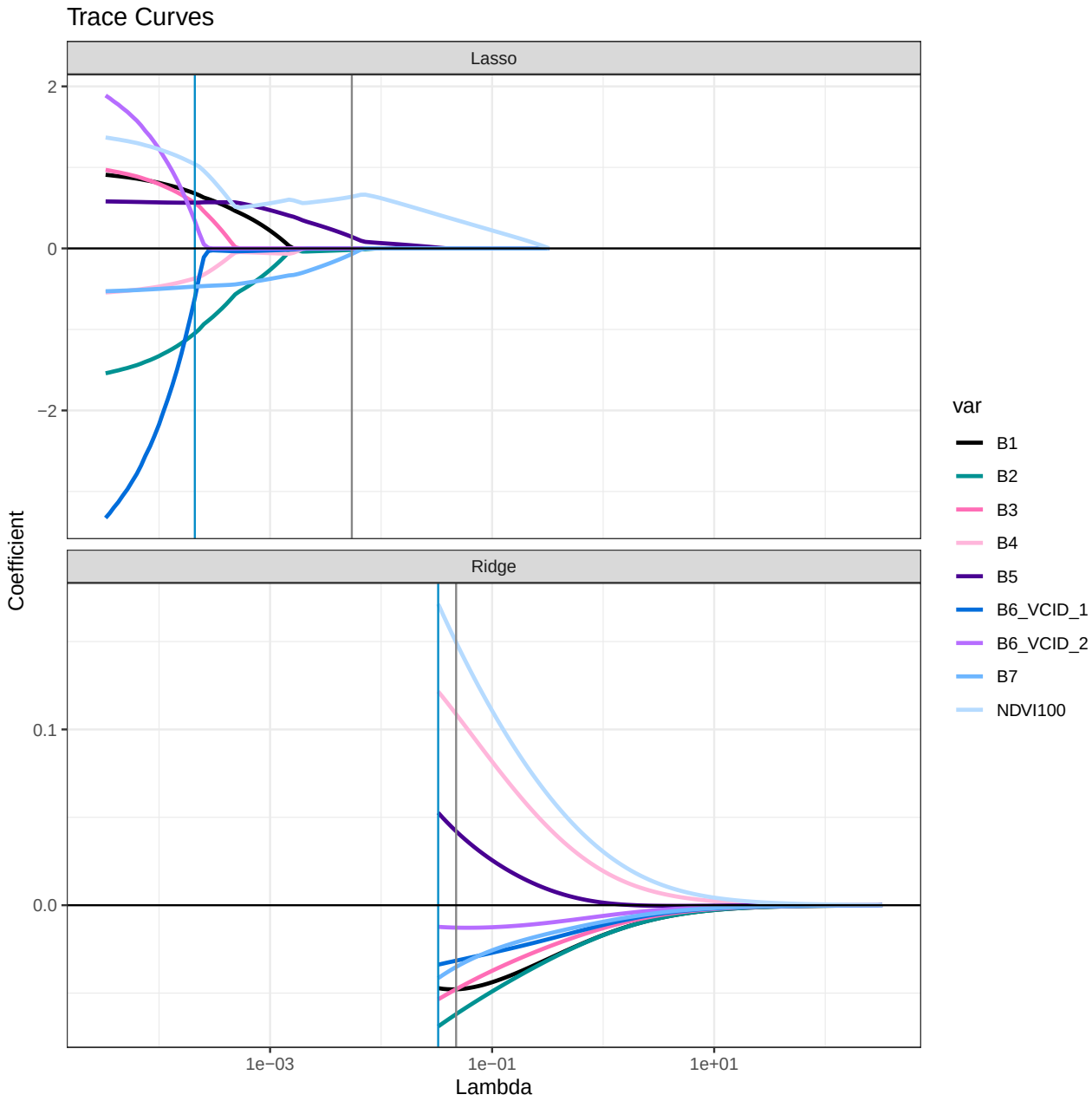
```
## # A tibble: 6 x 4
##   var   lambda     value model
##   <chr>  <dbl>     <dbl> <chr>
## 1 B1      327. -2.94e-38 Ridge
## 2 B1      298. -9.76e- 5 Ridge
## 3 B1      271. -1.07e- 4 Ridge
## 4 B1      247. -1.18e- 4 Ridge
## 5 B1      225. -1.29e- 4 Ridge
## 6 B1      205. -1.42e- 4 Ridge
```

Now we can plot the trace curves.

```
lambda.lines <- data.frame(model=c('Ridge', 'Lasso'),
                           lambda.min = c(m3$lambda.min, m4$lambda.min),
                           lambda.1se = c(m3$lambda.1se, m4$lambda.1se))
dg <- coefs1

g <- ggplot(data=dg, aes(x=lambda, y=value, group=var, color=var))+
    geom_line(alpha=1, linewidth=1)+
    facet_wrap(~model, ncol=1, scales='free_y')+
    geom_vline(data=lambda.lines, aes(xintercept=lambda.min),
               color = pubblue)+
    geom_vline(data=lambda.lines, aes(xintercept=lambda.1se),
               color = pubmediumgray)+
    scale_x_log10()+
    geom_hline(yintercept = 0)+
    scale_color_manual(values=cb.pal) +
    theme_bw() +
    labs(title='Trace Curves', x='Lambda', y='Coefficient')

g
```




```
ggsave('img/trace_curves.png', g, width=8, height=8)
```

It seems the coefficients for the Ridge model are much more stable than the ones for the Lasso model, as most of the coefficients in the Ridge model are close between -0.1 and 0.1 for every lambda value, but in the Lasso model, the coefficients are more spread out, from around -3 to 2. Also, for the Ridge model, we see that only 3 band values are positive for all values of Lambda, namely B5, B4, and NDVI100. For the Lasso model, we see that B3, B6_VCID_2, B1, and NDVI100 are positive, but B4 becomes negative.

Also, in the Lasso model, we see that most bands collapse to 0 as the values for Lambda increase. However, for NDVI100, it seems to decrease, then start following a linear pattern, and then decrease again. Additionally, in the Ridge model, it's interesting how most bands follow the same pattern, approaching 0 exponentially as Lambda increases, but B1 seems to first go farther away from 0 and then approach 0 exponentially like the other bands.

2. Cross-validation

Use cross-validation to make out of sample predictions for each observation and show the first 6 rows of the resulting data frame.

```
# for reproducibility
set.seed(123)

# set number of folds
k=5

# create a vector of fold numbers, repeating 1:k until reaching the number of rows in the data
folds <- rep(1:k, length.out = nrow(dm))

# create a new column in the data frame that randomly assigns a fold to each row without replacement
# (so that we don't have a row assigned to two or more folds)
dm$fold <- sample(folds,
                  nrow(dm),
                  replace = F)

# view the first few rows of the data frame to see the fold assignments
head(dm)
```

```
## # A tibble: 6 x 15
##      B1    B2    B3    B4    B5 B6_VCID_1 B6_VCID_2    B7 NDVI100   veg   predm1
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl> <dbl>    <dbl>
## 1  80.4  69.2  73.4  91.4  79        130.      148.  51.3   15.1      1 1.00
## 2  83.5  73.5  78.9  94.6  87.1      131.      150   57.9   13.4      1 1.00
## 3  91.0  81.7  95.0  97.1 111.       131.      149.  79.2    4.71     1 0.974
## 4  87.9  78.3  87.8  95.3 102.       131.      150.  71.1    7.67     1 0.997
## 5  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## 6  85.9  77.4  91.9  75.1  93.0      139.      164.  77.0   -9.53     0 0.000448
## # i 4 more variables: predm2 <dbl>, predm3 <dbl[,1]>, predm4 <dbl[,1]>,
## #   fold <int>
```

```
# create a data frame to keep track of the metrics
metrics <- data.frame(fold = 1:k,
                      llr1se = NA,
                      llrmin = NA,
                      lllas1se = NA,
                      lllasmin = NA,
                      llm1 = NA,
                      llm2 = NA)

# initialize the prediction columns
dm[, c('r.1se', 'las.1se', 'r.min', 'las.min', 'm1pred', 'm2pred')] = NA

# loop through the folds
for (j in 1:k){
  cat(j,'')

  ## train rows are the ones not in the j-th fold
  train.rows <- dm$fold != j

  # test rows are the observations in the j-th fold
  test.rows  <- dm$fold == j
```

```
## ======
## fit models to training data
## ======

## logistic regression with no regularization using only NDVI100
m1 <- glm(veg ~ NDVI100, data = dm[train.rows, 1:10], family = "binomial")

## logistic regression with no regularization using all band values
m2 <- glm(veg ~ ., data = dm[train.rows,1:10], family = "binomial")

## logistic regression with ridge regularization using all band values
r.train <- cv.glmnet(x = x[train.rows,], y = dm$veg[train.rows], family = 'binomial', alpha = 0)

## logistic regression with lasso regularization using all band values
las.train <- cv.glmnet(x = x[train.rows,], y = dm$veg[train.rows], family = 'binomial', alpha = 1)

## ======
## make predictions
## ======

dm$m1pred[test.rows] = predict(m1, newdata=dm[test.rows,], type='response')
dm$m2pred[test.rows] = predict(m2, newdata=dm[test.rows,], type='response')
dm$r.1se[test.rows] = predict(r.train, newx=x[test.rows,], s='lambda.1se', type='response')
dm$r.min[test.rows] = predict(r.train, newx=x[test.rows,], s='lambda.min', type='response')
dm$las.1se[test.rows] = predict(las.train, newx=x[test.rows,], s='lambda.1se', type='response')
dm$las.min[test.rows] = predict(las.train, newx=x[test.rows,], s='lambda.min', type='response')

## Test logloss for each fold
metrics[j,'llr1se'] = -mean(dm$veg[test.rows]*
                            log(dm$r.1se[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$r.1se[test.rows]))
metrics[j,'llrmin'] = -mean(dm$veg[test.rows]*
                            log(dm$r.min[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$r.min[test.rows]))
metrics[j,'lllas1se'] = -mean(dm$veg[test.rows]*
                            log(dm$las.1se[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$las.1se[test.rows]))
metrics[j,'lllasmin'] = -mean(dm$veg[test.rows]*
                            log(dm$las.min[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$las.min[test.rows]))
metrics[j,'llm1'] = -mean(dm$veg[test.rows]*
                            log(dm$m1pred[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$m1pred[test.rows]))
metrics[j,'llm2'] = -mean(dm$veg[test.rows]*
                            log(dm$m2pred[test.rows]) +
                            (1-dm$veg[test.rows])*
                            log(1-dm$m2pred[test.rows]))
```

```
}
```

```
## 1 2 3 4
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## 5
```

```
head(dm %>% select('r.1se', 'r.min', 'las.1se', 'las.min', 'm1pred', 'm2pred'))
```

```
## # A tibble: 6 x 6
##    r.1se  r.min  las.1se   las.min   m1pred        m2pred
##    <dbl>  <dbl>    <dbl>     <dbl>    <dbl>         <dbl>
## 1 0.994  0.996  1.00     1.00      1.00     1
## 2 0.989  0.994  1.00     1.00      1.00     1.00
## 3 0.901  0.917  0.959    1.00      0.980    1.00
## 4 0.947  0.960  0.993    0.999     0.996    0.999
## 5 0.0447 0.0313 0.000544 0.000104  0.000384 0.000113
## 6 0.0619 0.0349 0.00173  0.0000470 0.000357 0.00000000166
```

```
metrics
```

```
##   fold    llr1se    llrmin   lllas1se   lllasmin       llm1       llm2
## 1    1 0.1669045 0.1498231 0.09445221 0.09269673 0.09517429 0.11472190
## 2    2 0.1395895 0.1235677 0.06310080 0.04475571 0.06630667 0.06393426
## 3    3 0.1757241 0.1656531 0.10224240 0.06953002 0.14931390 0.05261686
## 4    4 0.2606668 0.2392228 0.19783758 0.35470053 0.21519616 0.93120602
## 5    5 0.1665322 0.1576225 0.13905100 0.09379078 0.17305322 0.09587438
```

3. Log Loss

Compute log loss for all models. Compare across models, compare in-sample vs CV log loss, and discuss any notable observations.

```
logloss <- data.frame(m1 = NA,
                      m2 = NA,
                      m3 = NA,
                      m4 = NA,
                      m1out = NA,
                      m2out = NA,
                      r1se = NA,
                      rmin = NA,
                      las1se = NA,
                      lasmin = NA)

# Log loss of every model (in sample)
logloss$m1 <- (-mean(dm$veg*log(dm$predm1) + (1-dm$veg)*log(1-dm$predm1)))
logloss$m2 <- (-mean(dm$veg*log(dm$predm2) + (1-dm$veg)*log(1-dm$predm2)))
logloss$m3 <- (-mean(dm$veg*log(dm$predm3) + (1-dm$veg)*log(1-dm$predm3)))
logloss$m4 <- (-mean(dm$veg*log(dm$predm4) + (1-dm$veg)*log(1-dm$predm4)))

# Log Loss of every model (out of sample)
logloss$r1se <- (-mean(dm$veg*log(dm$r.1se) + (1-dm$veg)*log(1-dm$r.1se)))
logloss$rmin <- (-mean(dm$veg*log(dm$r.min) + (1-dm$veg)*log(1-dm$r.min)))
logloss$las1se <- (-mean(dm$veg*log(dm$las.1se) + (1-dm$veg)*log(1-dm$las.1se)))
logloss$lasmin <- (-mean(dm$veg*log(dm$las.min) + (1-dm$veg)*log(1-dm$las.min)))
logloss$m1out <- (-mean(dm$veg*log(dm$m1pred) + (1-dm$veg)*log(1-dm$m1pred)))
```

```
logloss$m2out <- (-mean(dm$veg*log(dm$m2pred) + (1-dm$veg)*log(1-dm$m2pred)))

logloss <- logloss %>%
  pivot_longer(cols = everything(), names_to = 'model', values_to = 'logloss') %>%
  mutate(sample = ifelse(model %in% c('m1', 'm2', 'm3', 'm4'), 'in', 'out'))

logloss
```

```
## # A tibble: 10 x 3
##    model  logloss sample
##    <chr>    <dbl> <chr>
##  1 m1      0.132  in
##  2 m2      0.0683 in
##  3 m3      0.179  in
##  4 m4      0.107  in
##  5 m1out   0.140  out
##  6 m2out   0.252  out
##  7 r1se    0.182  out
##  8 rmin    0.167  out
##  9 las1se  0.119  out
## 10 lasmin  0.131  out
```

Across in-sample vs. out-sample, the log loss for Ridge is consistently higher than the log loss for Lasso (see m3 vs. m4, r1se vs. las1se, and rmin vs. lasmin).

It seems in-sample models have similar log loss to out-sample models (see m3 vs. r1se, m4 vs. las1se)

Also, it appears that the log loss for the in-sample logistic regression with no regularization has the lowest log loss.

It seems choosing lambda.min is better for both Ridge and Lasso models, as the log loss is lower for lambda.min in both models.