Ensemble Methods for Classification

Majority Vote and Stacking

Ensembles Example: Ionosphere radar data

This example is adapted from a discussion at https://burakhimmetoglu.com/2016/12/01/stacking-models-for-improved-predictions/

Our data set for today is published and described at https://archive.ics.uci.edu/ml/datasets/ionosphere:

> This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. See the paper for more details. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.
>
> Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this databse are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.
>
> Attribute Information:
>
> - All 34 are continuous
> - The 35th attribute is either "good" or "bad" according to the definition summarized above. This is a binary classification task.

```r
library(readr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(purrr)
library(glmnet)
library(caret)

# read in data
ionosphere <- read_csv("http://www.evanlray.com/data/UCIML/ionosphere/ionosphere.data", col_names = FALSE)

# X2 was all 0's
ionosphere <- ionosphere %>% select(-X2)

# Convert prediction target to a factor
ionosphere$X35 <- factor(ionosphere$X35)


## Initial train/test split ("estimation"/test) and cross-validation folds
set.seed(63770)
tt_inds <- caret::createDataPartition(ionosphere$X35, p = 0.8)
ionosphere_train <- ionosphere %>% slice(tt_inds[[1]])
ionosphere_test <- ionosphere %>% slice(-tt_inds[[1]])

crossval_val_fold_inds <- caret::createFolds(
  y = ionosphere_train$X35, # response variable as a vector
  k = 10 # number of folds for cross-validation
)

get_complementary_inds <- function(x) {
  return(seq_len(nrow(ionosphere_train))[-x])
}
crossval_train_fold_inds <- map(crossval_val_fold_inds, get_complementary_inds)
```

Individual Methods

Logistic Regression

```
logistic_fit <- train(
  form = X35 ~ .,
  data = ionosphere_train,
  family = "binomial", # this is an argument to glm
  method = "glm", # method for fit
  trControl = trainControl(method = "cv", # evaluate method performance via cross-validation
    number = 10, # number of folds for cross-validation
    index = crossval_train_fold_inds, # I'm specifying which folds to use, for consistency across methods
    indexOut = crossval_val_fold_inds, # I'm specifying which folds to use, for consistency across methods
    returnResamp = "all", # return information from cross-validation
    savePredictions = TRUE, # return validation set predictions from cross-validation
    classProbs = TRUE) # return validation set predicted class probabilities from cross-validation
)

logistic_fit$results
```

```
##   parameter  Accuracy     Kappa AccuracySD   KappaSD
## 1      none 0.8504926 0.6652007 0.07110272 0.1659337
```

```
head(logistic_fit$pred)
```

```
##   pred obs            b            g rowIndex parameter Resample
## 1    g   g 2.052430e-01 7.947570e-01        1      none   Fold01
## 2    g   g 2.576157e-02 9.742384e-01       10      none   Fold01
## 3    g   b 3.994962e-06 9.999960e-01       13      none   Fold01
## 4    g   g 3.042912e-04 9.996957e-01       16      none   Fold01
## 5    g   g 1.427377e-03 9.985726e-01       17      none   Fold01
## 6    b   b 1.000000e+00 1.735932e-11       38      none   Fold01
```

```
logistic_fit$pred %>%
  group_by(Resample) %>%
  summarize(accuracy = mean(pred == obs)) %>%
  ungroup() %>%
  summarize(accuracy = mean(accuracy))
```

```
## # A tibble: 1 x 1
##   accuracy
##      <dbl>
## 1    0.850
```

KNN

```
knn_fit <- train(
  form = X35 ~ .,
  data = ionosphere_train,
  method = "knn",
  preProcess = "scale",
  trControl = trainControl(method = "cv",
    number = 10,
    index = crossval_train_fold_inds, # I'm specifying which folds to use, for consistency across methods
    indexOut = crossval_val_fold_inds, # I'm specifying which folds to use, for consistency across methods
    returnResamp = "all",
    savePredictions = TRUE,
    classProbs = TRUE),
  tuneGrid = data.frame(k = 1:20)
)

knn_fit$results
```

```
##     k  Accuracy     Kappa AccuracySD   KappaSD
## 1   1 0.8540640 0.6570214 0.08983538 0.2252713
## 2   2 0.8362069 0.6136325 0.07200427 0.1706708
## 3   3 0.8147783 0.5538864 0.09519951 0.2273648
## 4   4 0.8041872 0.5226215 0.09867257 0.2402436
## 5   5 0.8219212 0.5680881 0.09692917 0.2403482
## 6   6 0.8254926 0.5792690 0.08680310 0.2128852
## 7   7 0.8040640 0.5219281 0.06620901 0.1715432
## 8   8 0.8076355 0.5324729 0.06163325 0.1578542
## 9   9 0.8041872 0.5237628 0.05671862 0.1452014
## 10 10 0.8075123 0.5300637 0.06400532 0.1680257
## 11 11 0.7932266 0.4880301 0.05349060 0.1460922
## 12 12 0.7896552 0.4784264 0.04981489 0.1364785
## 13 13 0.8003695 0.5071866 0.05702811 0.1567114
## 14 14 0.8003695 0.5082105 0.05182000 0.1414887
## 15 15 0.7896552 0.4785748 0.06466996 0.1707640
## 16 16 0.7754926 0.4389018 0.05922736 0.1593515
## 17 17 0.7754926 0.4383273 0.06157375 0.1677752
## 18 18 0.7684729 0.4133543 0.07438714 0.2072648
## 19 19 0.7647783 0.4022990 0.07254520 0.2019538
## 20 20 0.7472906 0.3506893 0.04908562 0.1432625
```

Trees

```
rpart_fit <- train(
  form = X35 ~ .,
  data = ionosphere_train,
  method = "rpart",
  trControl = trainControl(method = "cv",
    number = 10,
    index = crossval_train_fold_inds, # I'm specifying which folds to use, for consistency across methods
    indexOut = crossval_val_fold_inds, # I'm specifying which folds to use, for consistency across methods
    returnResamp = "all",
    savePredictions = TRUE,
    classProbs = TRUE),
  tuneGrid = data.frame(cp = seq(from = 0, to = 1, length = 20))
)

rpart_fit$results
```

```
##            cp  Accuracy      Kappa  AccuracySD    KappaSD
## 1  0.00000000 0.8543103 0.67345204 0.058505963 0.13243804
## 2  0.05263158 0.8828818 0.74337870 0.057119643 0.12038235
## 3  0.10526316 0.8828818 0.74337870 0.057119643 0.12038235
## 4  0.15789474 0.8828818 0.74337870 0.057119643 0.12038235
## 5  0.21052632 0.8364532 0.61862568 0.069272228 0.19298353
## 6  0.26315789 0.7795567 0.46022616 0.045605318 0.11811999
## 7  0.31578947 0.7795567 0.46022616 0.045605318 0.11811999
## 8  0.36842105 0.7795567 0.46022616 0.045605318 0.11811999
## 9  0.42105263 0.7795567 0.46022616 0.045605318 0.11811999
## 10 0.47368421 0.7795567 0.46022616 0.045605318 0.11811999
## 11 0.52631579 0.6477833 0.02432432 0.024382992 0.07692027
## 12 0.57894737 0.6406404 0.00000000 0.007009975 0.00000000
## 13 0.63157895 0.6406404 0.00000000 0.007009975 0.00000000
## 14 0.68421053 0.6406404 0.00000000 0.007009975 0.00000000
## 15 0.73684211 0.6406404 0.00000000 0.007009975 0.00000000
## 16 0.78947368 0.6406404 0.00000000 0.007009975 0.00000000
## 17 0.84210526 0.6406404 0.00000000 0.007009975 0.00000000
## 18 0.89473684 0.6406404 0.00000000 0.007009975 0.00000000
## 19 0.94736842 0.6406404 0.00000000 0.007009975 0.00000000
## 20 1.00000000 0.6406404 0.00000000 0.007009975 0.00000000
```

Test set predictions from each of the 3 methods above:

```
logistic_preds <- predict(logistic_fit, newdata = ionosphere_test)
mean(ionosphere_test$X35 != logistic_preds)
```

```
## [1] 0.07142857
```

```
knn_preds <- predict(knn_fit, newdata = ionosphere_test)
mean(ionosphere_test$X35 != knn_preds)
```

```
## [1] 0.08571429
```

```
rpart_preds <- predict(rpart_fit, newdata = ionosphere_test)
mean(ionosphere_test$X35 != rpart_preds)
```

```
## [1] 0.05714286
```

Ensemble Methods

Majority Vote

```r
majority_vote_preds <- ifelse(
  (logistic_preds == "g") + (knn_preds == "g") + (rpart_preds == "g") >= 2,
  "g",
  "b"
)

mean(ionosphere_test$X35 != majority_vote_preds)
```

```
## [1] 0.02857143
```

Mean of Class Probabilities from Individual Methods

```r
logistic_class_probs <- predict(logistic_fit, newdata = ionosphere_test, type = "prob")
logistic_prob_g <- logistic_class_probs[, 2]

knn_class_probs <- predict(knn_fit, newdata = ionosphere_test, type = "prob")
knn_prob_g <- knn_class_probs[, 2]

rpart_class_probs <- predict(rpart_fit, newdata = ionosphere_test, type = "prob")
rpart_prob_g <- rpart_class_probs[, 2]

mean_prob_g <- (logistic_prob_g + knn_prob_g + rpart_prob_g) / 3
mean_prob_preds <- ifelse(
  mean_prob_g >= 0.5,
  "g",
  "b"
)
mean(ionosphere_test$X35 != mean_prob_preds)
```

```
## [1] 0.04285714
```

Stacking: Fit a model to combine predicted class membership probabilities

Process:

Estimation:

1. Get cross-validated predictions for each "stage 1" or "component" model (this was done above)
2. Create a new data set where the explanatory variables are the cross-validated predictions from the component models
3. Fit a "stage 2" model to predict the response based on the component model predictions

Prediction for test set:

4. Re-fit each component model to the full training data set, make predictions for the test set from each component model (this was done above)
5. Create a new data set where the explanatory variables are the test set predictions from the component models
6. Predict using the stage 2 model fit from step 3 and the data frame created in step 5.

```r
# Step 2: data set with component model predictions as explanatory variables
ionosphere_train <- ionosphere_train %>%
  mutate(
    logistic_prob_g = logistic_fit$pred %>%
      arrange(rowIndex) %>%
      pull(g),
    knn_prob_g = knn_fit$pred %>%
      filter(k == 1) %>%
      arrange(rowIndex) %>%
      pull(g),
    rpart_prob_g = rpart_fit$pred %>%
      filter(cp == rpart_fit$bestTune$cp) %>%
      arrange(rowIndex) %>%
      pull(g)
  )

# Step 3: fit model using component model predictions as explanatory variables
stacking_logistic_fit <- train(
  form = X35 ~ logistic_prob_g + knn_prob_g + rpart_prob_g,
  data = ionosphere_train,
  family = "binomial",
  method = "glm"
)

# Step 5: Assemble data frame of test set predictions from each component model
stacking_test_x <- data.frame(
  logistic_prob_g = logistic_prob_g,
  knn_prob_g = knn_prob_g,
  rpart_prob_g = rpart_prob_g
)

# Step 6: Stacked model predictions
stacking_preds <- predict(stacking_logistic_fit, stacking_test_x)

# Calculate error rate
mean(ionosphere_test$X35 != stacking_preds)
```

```
## [1] 0.01428571
```

Stacking via KNN

- We could also use other methods for the second stage model.

```
ionosphere_train <- ionosphere_train %>%
  mutate(
    logistic_prob_g = logistic_fit$pred %>%
      arrange(rowIndex) %>%
      pull(g),
    knn_prob_g = knn_fit$pred %>%
      filter(k == 1) %>%
      arrange(rowIndex) %>%
      pull(g),
    rpart_prob_g = rpart_fit$pred %>%
      filter(cp == rpart_fit$bestTune$cp) %>%
      arrange(rowIndex) %>%
      pull(g)
  )

stacking_knn_fit <- train(
  form = X35 ~ logistic_prob_g + knn_prob_g + rpart_prob_g,
  data = ionosphere_train,
  method = "knn"
)

# Assemble data frame of test set predictions from each
# component model (these were obtained in the previous part)
stacking_test_x <- data.frame(
  logistic_prob_g = logistic_prob_g,
  knn_prob_g = knn_prob_g,
  rpart_prob_g = rpart_prob_g
)

# Stacked model predictions
stacking_preds <- predict(stacking_knn_fit, stacking_test_x)
mean(ionosphere_test$X35 != stacking_preds)
```

```
## [1] 0.01428571
```

Notes about relative performance of these methods

- The results with the seed I have set make the stacking approaches look amazing - but in different runs I did as I was developing this, relative performance of the ensemble approaches here varied.
- In general, stacking is the best of these ensemble approaches in terms of expected value of performance, if the methods' performance is not all equal and we have enough training set data.
- Note that in this example we had only 3 stage 1/component models. In general, ensembles are most useful when we have:
  - A large number of models that are "diverse"/uncorrelated/predictions are close to independent
  - Enough training data available to reliably tell which component models are best and how to combine their predictions effectively.
- Formally, the stacking procedure I did here is not quite right:
  - You shouldn't use the same cross-validation results both to select tuning parameters like K for KNN and cp for classification trees, AND as inputs to stacking.
  - Doing this means models that overfit validation data will get too much weight
  - In practice if you're selecting one tuning parameter this shouldn't matter too much.