BIOS 635: Learning with Unbalanced Data

Kevin Donovan

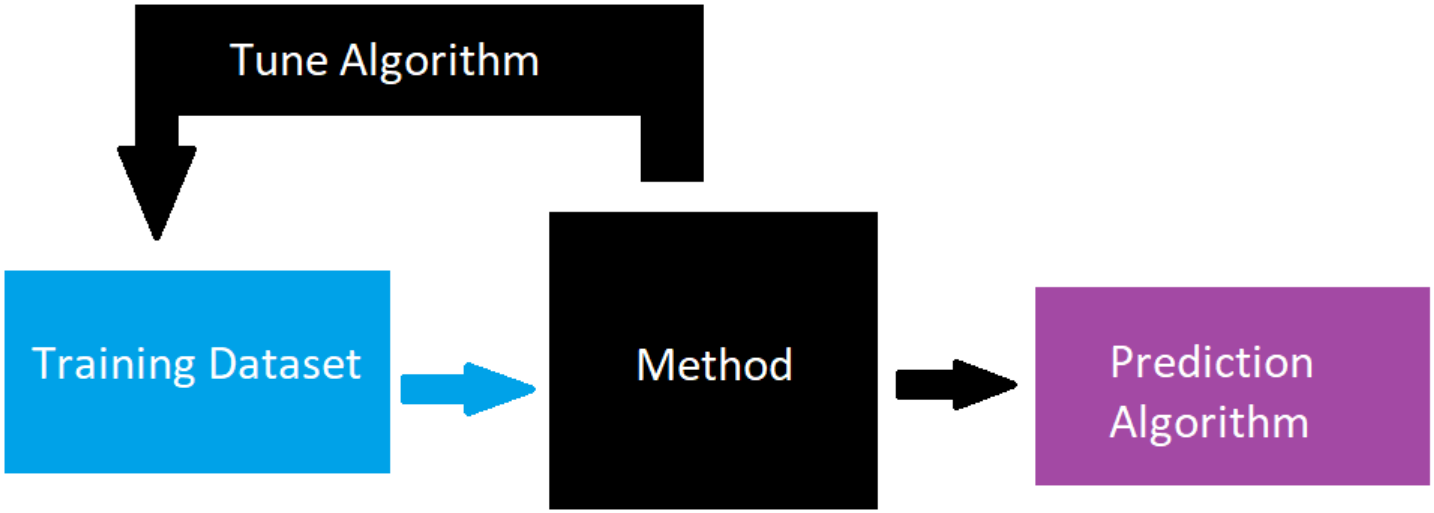
4/8/2021

Review

- Homework 7 due on Saturday by 11PM EST

Supervised Learning

- **Recall**: Train algorithm based on data with observed classes
- Training done based on minimizing error/maxing accuracy
- $\rightarrow$ driven by **majority class** if large imbalance



Imbalanced Data

- **Recall**: IBIS dataset
- Low number of infants with autism spectrum disorder (ASD)
- **Still important to predict this group well** (cost of missing them is high!)

```
freq(ibis_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## ibis_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg    217     76.14     76.14
##       HR_ASD     68     23.86     23.86
##         <NA>      0                0.00
```

Analytic Methods

- From homework, know that low frequency class can be tough

```
tt_indices <- createDataPartition(y=ibis_data$asd_group, p=0.6, list = FALSE)

train_data <- ibis_data[tt_indices,]
test_data <- ibis_data[-tt_indices,]

freq(train_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## train_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg    131     76.16     76.16
##       HR_ASD     41     23.84     23.84
##         <NA>      0                0.00
```

```
freq(test_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## test_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
```

```
## ------------ ------ --------- ---------
##       HR_Neg     86     76.11     76.11
##       HR_ASD     27     23.89     23.89
##         <NA>      0                0.00
```

```
# Use random forest for example
rf_fit <- randomForest(formula = asd_group~., data = train_data)

test_predict <- predict(rf_fit, newdata = test_data)

# Confusion matrix
confusionMatrix(data=test_predict, reference=test_data$asd_group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction HR_ASD HR_Neg
##     HR_ASD      1      1
##     HR_Neg     26     85
##
##                Accuracy : 0.7611
##                  95% CI : (0.6717, 0.8363)
##     No Information Rate : 0.7611
##     P-Value [Acc > NIR] : 0.5514
##
##                   Kappa : 0.0372
##
##  Mcnemar's Test P-Value : 3.86e-06
##
##             Sensitivity : 0.03704
##             Specificity : 0.98837
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.76577
```

```
##              Prevalence : 0.23894
##          Detection Rate : 0.00885
##    Detection Prevalence : 0.01770
##       Balanced Accuracy : 0.51270
##
##        'Positive' Class : HR_ASD
##
```

Analytic Methods

- What to do?
- Main methods:
    1. *Weighting observations*
    2. *Super-sampling*

Weighting

- By default, each obs gets same weight $\rightarrow$ contribute to error the same amount
- But, **error in predicting one type may have higher cost then for other type**
- Can implement this using chosen weights
- Ex. linear regression
  - *No weights*

$$\hat{y} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta x_i)^2$$

- Weights

$$\hat{y}_w = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} w_i (y_i - \beta x_i)^2$$

$$\text{where } \{w_i\}_{i=1}^{n} \text{ are subject-specific weights}$$

- Common choice for weights: *inverse probability weighting*

$$w_i = 1/\hat{\pi}_k \text{ for } k = 1, \ldots, K$$

$$\text{where } \hat{\pi}_k = \sum_{i=1}^{n} I(i \text{ in group } K)/n$$

- Those in rare groups get **high weight**, cost for their error is high

Weighting

- With `caret` can implement in `train` function (`weights` argument)

```
tt_indices <- createDataPartition(y=ibis_data$asd_group, p=0.6, list = FALSE)

ibis_data <- data.frame(lapply(ibis_data, as.factor))

train_data <- ibis_data[tt_indices,]
test_data <- ibis_data[-tt_indices,]

freq(train_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## train_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg    131     76.16     76.16
##       HR_ASD     41     23.84     23.84
##         <NA>      0                0.00
```

```
freq(test_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## test_data$asd_group
## Type: Factor
```

```
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg     86     76.11     76.11
##       HR_ASD     27     23.89     23.89
##         <NA>      0                0.00
```

```
# Compute weights
freq_table <- freq(train_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
props <- freq_table[,"% Total"]
props <- ifelse(train_data$asd_group=="HR_Neg", props["HR_Neg"]/100,
                props["HR_ASD"]/100)
model_weights <- 1/props
ftable(round(model_weights, 3))
```

```
##  1.313 4.195
##
##    131    41
```

```
ftable(train_data$asd_group)
```

```
##  HR_ASD HR_Neg
##
##      41    131
```

```
# Use random forest for example
rf_fit <- train(asd_group~., data = train_data,
                      method = "rf", weights = model_weights,
               trControl=trainControl(method="cv", number=2))
```

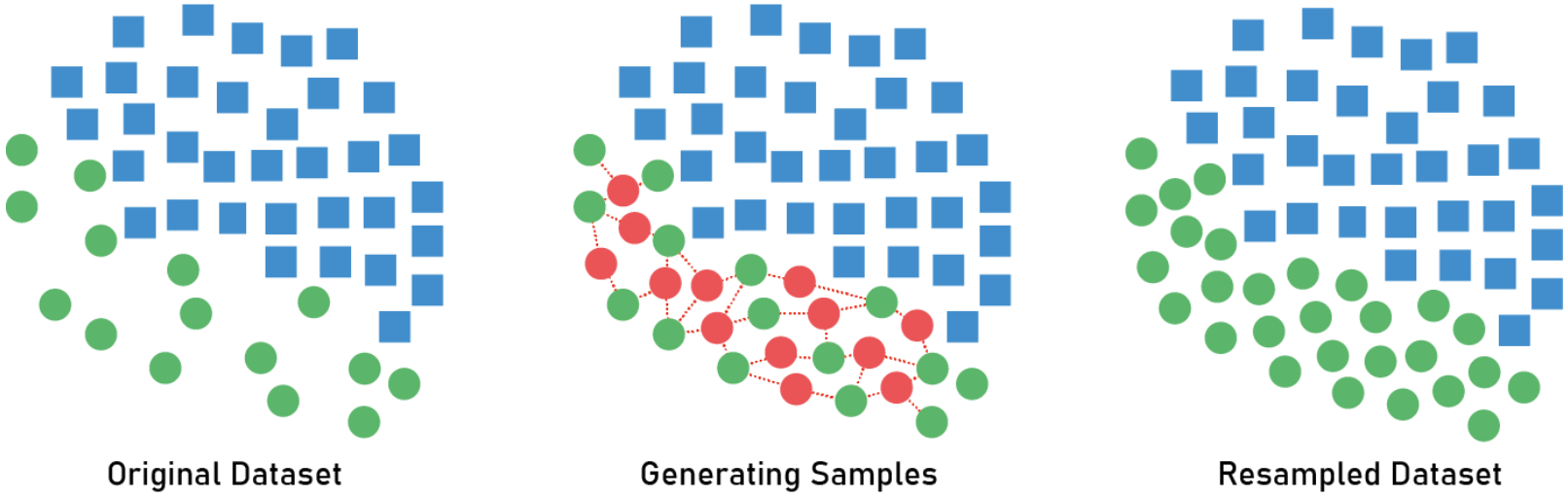
```r
test_predict <- predict(rf_fit, newdata = test_data)

# Confusion matrix
confusionMatrix(data=test_predict, reference=test_data$asd_group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction HR_ASD HR_Neg
##     HR_ASD      8      2
##     HR_Neg     19     84
##
##                Accuracy : 0.8142
##                  95% CI : (0.7301, 0.8811)
##     No Information Rate : 0.7611
##     P-Value [Acc > NIR] : 0.1104572
##
##                   Kappa : 0.3483
##
##  Mcnemar's Test P-Value : 0.0004803
##
##             Sensitivity : 0.2963
##             Specificity : 0.9767
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8155
##              Prevalence : 0.2389
##          Detection Rate : 0.0708
##    Detection Prevalence : 0.0885
##       Balanced Accuracy : 0.6365
##
##        'Positive' Class : HR_ASD
##
```

Supersampling

- Second idea: up/downsampling
- **Concept**: Why don't we up-sample the minority class (down-sample the majority)
  - ***Note**: Still need predictor distributions in synthetic training set to be maintained*
    - Why?
  - ***Leave test set alone, only create a synthetic sample from the training set***
    - Why?
- Most common method: SMOTE

Synthetic Minority Oversampling Technique



Supersampling

- Common method: `DmWR` package
- **However**, package has been removed from CRAN
- Alternatives:

```
tt_indices <- createDataPartition(y=ibis_data$asd_group, p=0.6, list = FALSE)

ibis_data <- data.frame(lapply(ibis_data, as.factor))

train_data <- ibis_data[tt_indices,]
test_data <- ibis_data[-tt_indices,]

freq(train_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## train_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg    131     76.16     76.16
##       HR_ASD     41     23.84     23.84
##         <NA>      0                0.00
```

```
freq(test_data$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

```
## Frequencies
## test_data$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_Neg     86     76.11     76.11
##       HR_ASD     27     23.89     23.89
##         <NA>      0                0.00
```

```
# SMOTE training set AFTER creating train and test sets
train_smote <- SMOTE(asd_group~., data=train_data, perc.under = 150)
freq(train_smote$asd_group, order="freq", totals = FALSE, cumul =FALSE)
```

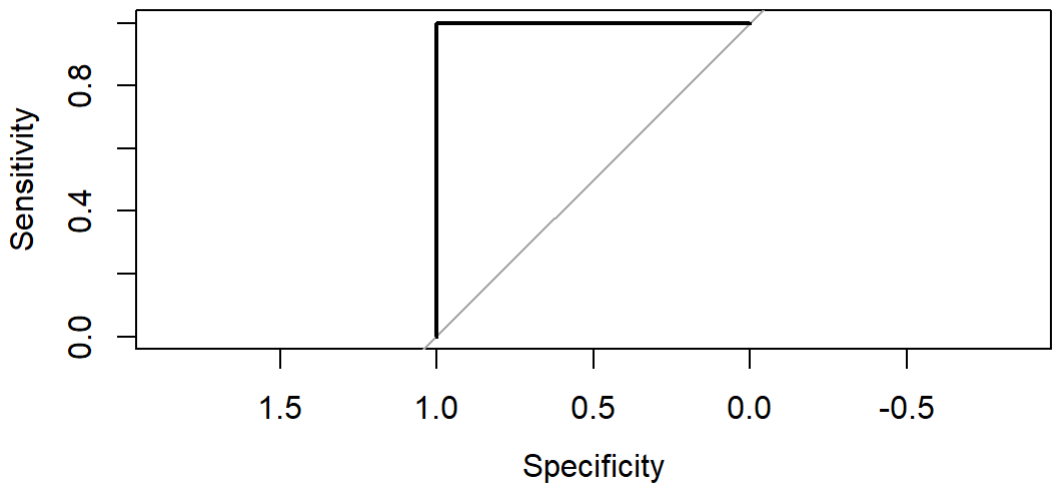
```
## Frequencies
## train_smote$asd_group
## Type: Factor
##
##                Freq   % Valid   % Total
## ------------ ------ --------- ---------
##       HR_ASD    123     50.00     50.00
##       HR_Neg    123     50.00     50.00
##         <NA>      0                0.00
```

```
# Use random forest for example
rf_fit <- train(asd_group~., data = train_smote,
                     method = "rf",
                trControl=trainControl(method="cv", number=2))
```

```r
test_predict <- predict(rf_fit, newdata = test_data)

# Confusion matrix
confusionMatrix(data=test_predict, reference=test_data$asd_group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction HR_ASD HR_Neg
##     HR_ASD     17     23
##     HR_Neg     10     63
##
##                Accuracy : 0.708
##                  95% CI : (0.615, 0.7897)
##     No Information Rate : 0.7611
##     P-Value [Acc > NIR] : 0.92170
##
##                   Kappa : 0.3108
##
##  Mcnemar's Test P-Value : 0.03671
##
##             Sensitivity : 0.6296
##             Specificity : 0.7326
##          Pos Pred Value : 0.4250
##          Neg Pred Value : 0.8630
##              Prevalence : 0.2389
##          Detection Rate : 0.1504
##    Detection Prevalence : 0.3540
##       Balanced Accuracy : 0.6811
##
##        'Positive' Class : HR_ASD
##
```

```
# Try changing prob threshold using ROC
train_roc <- roc(predictor=predict(rf_fit, newdata=train_smote, type="prob")[,1],
                 response=train_smote$asd_group)
plot(train_roc)
```




```
roc_thresh <- coords(train_roc, "best", "threshold")
test_predict_probs <- predict(rf_fit, newdata = test_data, type="prob")[,1]
test_predict_edit <- factor(ifelse(test_predict_probs>roc_thresh$threshold, "HR_ASD", "HR_Neg"))
confusionMatrix(data=test_predict_edit, reference=test_data$asd_group)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction HR_ASD HR_Neg
##     HR_ASD     17     23
##     HR_Neg     10     63
##
```

```
##                Accuracy : 0.708
##                  95% CI : (0.615, 0.7897)
##     No Information Rate : 0.7611
##     P-Value [Acc > NIR] : 0.92170
##
##                   Kappa : 0.3108
##
##  Mcnemar's Test P-Value : 0.03671
##
##             Sensitivity : 0.6296
##             Specificity : 0.7326
##          Pos Pred Value : 0.4250
##          Neg Pred Value : 0.8630
##              Prevalence : 0.2389
##          Detection Rate : 0.1504
##    Detection Prevalence : 0.3540
##       Balanced Accuracy : 0.6811
##
##        'Positive' Class : HR_ASD
##
```

Supersampling

- Caveats:
  - *SMOTE requires **some** numeric predictors to determine "neighborhoods" to draw samples*
    - **Ad hoc**: convert categorical to numeric
    - May be suboptimal (**Why?**)
  - *Can be difficult to select ROC-based threshold from training set after SMOTE*
    - Sometimes threshold chosen from test set, **but this is suboptimal!**