An Introduction to Ensemble Methods for Machine Learning

K. Tyler Wilcox
Rochester Institute of Technology

April 23, 2016

Classifying Spam Email: Isn't One Model Enough?

- Consider predicting which emails are spam (`spam` data from R package `kernlab`)
- $n = 4601$ observations
- $p = 57$ features
- $Y = \{spam, nonspam\}$
- No missing values to deal with
- Pre-process by dropping feature "num857" due to multicollinearity
- Standardize all remaining $p = 56$ features

Feature Engineering

```
# Prepare packages and spam data
pacman::p_load(kernlab, dplyr, caret, rpart)
data(spam)
spam = spam %>% tbl_df()
# Find correlated predictors
spam %>% select(-type) %>% cor() %>%
  findCorrelation(names = TRUE, verbose = TRUE)

## Compare row 32  and column  34 with corr  0.996
##   Means:  0.143 vs 0.059 so flagging column 32
## All correlations <= 0.9

## [1] "num857"
```

Training/Testing Split

```
# Drop highly redundant feature 'num857'
clean_spam = spam %>% select(-num857)
# Split data into train and test set
set.seed(122)
id_tr = clean_spam %>% select(type) %>% unlist() %>%
  createDataPartition(p = 0.75, list = FALSE)
train = clean_spam %>% dplyr::slice(id_tr)
test = clean_spam %>% dplyr::slice(-id_tr)
# Center and standarize training data
trans = train %>%
  preProcess(method = c("center", "scale"))
tr = trans %>% predict(newdata = train)
te = trans %>% predict(newdata = test)
```

Classification Trees (Breiman, 1984)

- Given data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- $x_i \in \mathcal{X}^p$, $y_i \in \{1, \ldots, g\}$, $i = 1, \ldots, n$
- A tree T partitions $\mathcal{X}$ into K regions $R_1, \ldots, R_K$ where

$$T = \cup_{k=1}^{K} R_k$$

- Tree built using a recursive partitioning algorithm that forms splits over a given feature x_j, $j = 1, \ldots, p$ to minimize node impurity (e.g., Gini index, cross entropy)
- The piecewise estimated function is then

$$\hat{f}_T(x) = \sum_{k=1}^{K} \hat{y}_k I_k(x) \text{ where } I_k(x) = 1_{x \in R_k}$$

$$\hat{y}_k = \arg\max \left\{ \frac{1}{|R_k|} \sum_{x_i \in R_k} I(y_i = c) \right\}, \; c = 1, \ldots, g$$

Visualizing Tree Partitioning

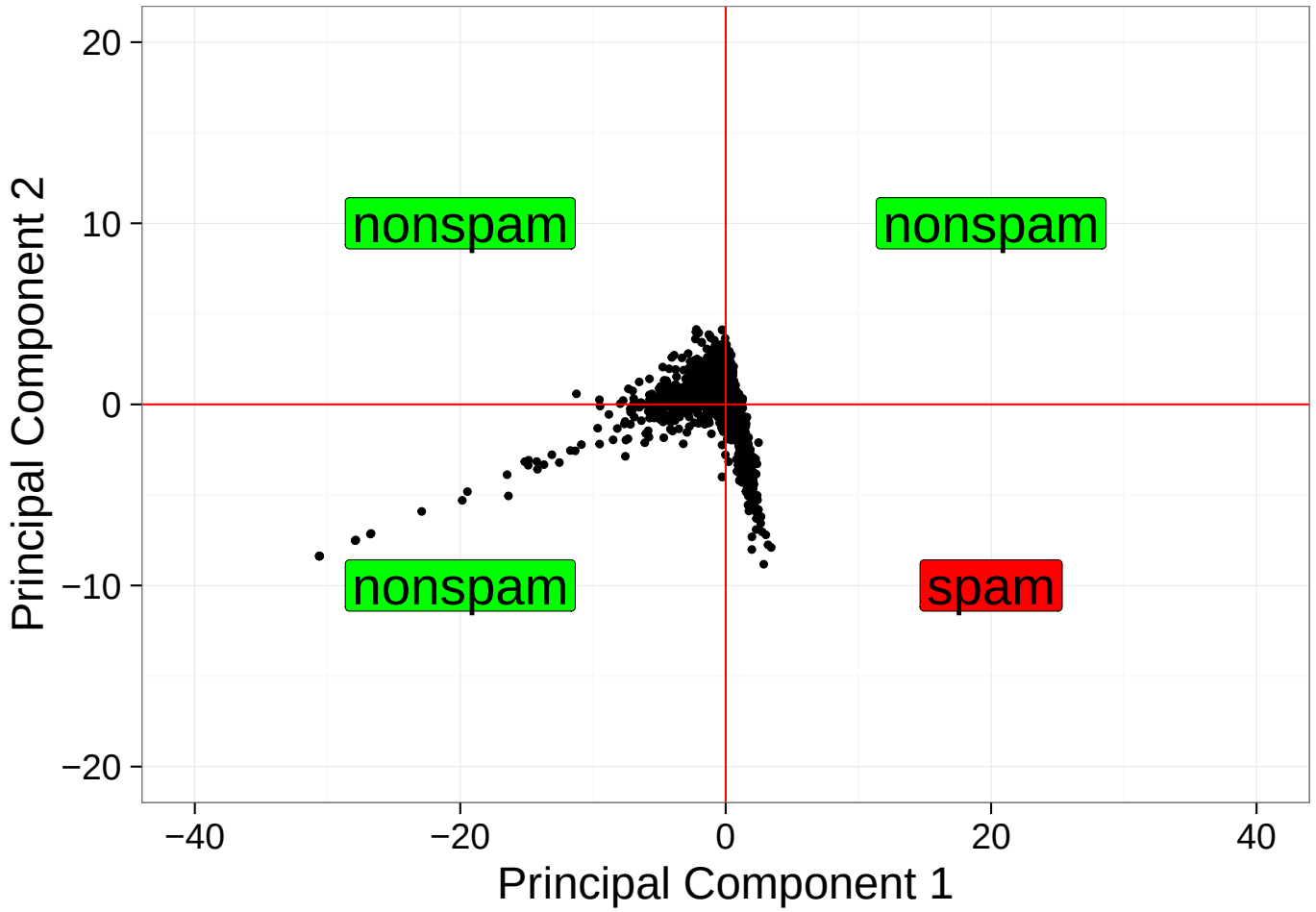

Figure 1. Classification tree estimated predictions

Classification Tree in R

```
# Set training options
ctrl = trainControl(method = "repeatedcv")
set.seed(333)
# Train and optimize tree, predict on test set
tree_tr = train(form = type ~ .,
                data = tr,
                trControl = ctrl,
                method = "rpart")
pr_tree = predict(tree_tr,
                  newdata = te %>% select(-type))
```

Classification Tree Accuracy: Can We Do Better?

- The pruned classification tree does reasonably well
- Accuracy = 87.04%

	Predictions	
True Response	nonspam	spam
nonspam	653	44
spam	105	348

Table 1. Confusion matrix for classification tree

Bias and Variance: Improving Accuracy

$$
\begin{aligned}
\text{Error}(x) &= \mathbb{E}[(f(x) - \hat{f}(x))^2] \\
&= \left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2 + \mathbb{E}\left[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2\right] + \sigma^2 \\
&= \text{Bias}^2 + \text{Variance} + \text{Noise}
\end{aligned}
$$

- Minimize bias $\rightarrow$ under-fit
- Minimize variance $\rightarrow$ over-fit
- Classification tree is fairly accurate but highly unstable
- Increase depth $\rightarrow$ reduce bias, increase variance
- Decrease depth $\rightarrow$ reduce variance, increase variance

General Ensemble

- Form M estimators of the true function g

$$\{\hat{g}_1(x), \hat{g}_m(x), \hat{g}_M(x)\}$$

- Assign weights α_m for each learner $\hat{g}_m(x)$, $m = 1, \ldots, M$
- Form the ensemble estimator

$$\hat{g}_{ensemble}(x) = \sum_{m=1}^{M} \alpha_m \hat{g}_m(x)$$

Reducing Variance: Bagging

- Breiman (1996) proposed bootstrap aggregation to reduce variance of an estimator
- Bagging increases accuracy by decreasing variance
- Let learner $\hat{g}_m(x) = \hat{f}_T$ be a classification tree trained on the bootstrap sample m, $m = 1, \ldots, M$
- Set weights $\alpha_m = \frac{1}{M}$
- Form the bagged estimator

$$\begin{aligned}\hat{g}_{bag}(x) &= \sum_{m=1}^{M} \alpha_m \hat{g}_m(x) \\ &= \frac{1}{M} \sum_{m=1}^{M} \hat{f}_T^{(m)}(x)\end{aligned}$$

- Trees grown to maximum depth $\rightarrow$ minimal bias

Bagging in R

```
pacman::p_load(doParallel, rpart)
set.seed(333)
cl <- makeCluster(detectCores() - 1, port = 11999)
registerDoParallel(cl)
# Train and optimize tree, predict on test set
bag_tr = train(form = type ~ .,
               data = tr,
               trControl = ctrl,
               method = "treebag",
               allowParallel = TRUE)
stopCluster(cl)
saveRDS(object = bag_tr, file = "bag_train.rds")
pr_bag = predict(bag_tr,
                 newdata = te %>% select(-type))
```

Bagging Accuracy

- Bagging dramatically improves classification tree performance
- Accuracy = 95.91%

	Predictions	
True Response	nonspam	spam
nonspam	677	20
spam	27	426

Table 2. Confusion matrix for bagging

Bagging + Random Subspace Learning: Random Forests

- Breiman (2001) proposes random forest algorithm
- Random forest uses bootstrap sampling of both observations and features for each learner
- Let learner $\hat{g}_m(x) = \hat{f}_T$ be a classification tree trained on the bootstrap sample of observations m, $m = 1, \ldots, M$ using a random subset of features
- Set weights $\alpha_m = \frac{1}{M}$
- Form the random forest estimator

$$\begin{aligned}\hat{g}_{rF}(x) &= \sum_{m=1}^{M} \alpha_m \hat{g}_m(x) \\ &= \frac{1}{M} \sum_{m=1}^{M} \hat{f}_T^{(m)}(x)\end{aligned}$$

Random Forest in R

```
pacman::p_load(doParallel, ranger)
set.seed(333)
cl <- makeCluster(detectCores() - 1, port = 11999)
registerDoParallel(cl)
# Train and optimize tree, predict on test set
rf_tr = train(form = type ~ .,
              data = tr,
              trControl = ctrl,
              method = "ranger")
stopCluster(cl)
saveRDS(object = rf_tr, file = "ranger_train.rds")
pr_rf = predict(rf_tr,
                newdata = te %>% select(-type))
```

Random Forest Accuracy

- Random forest provides an improvement over bagging performance
- Accuracy = 96.26%

	Predictions	
True Response	nonspam	spam
nonspam	678	19
spam	24	429

Table 3. Confusion matrix for random forest

Weighted Ensembles: AdaBoost

- Freund and Schapire (1995) propose AdaBoost
- Weak learners aggregated with updated weights
- Bootstrapping distribution for learner g_m depends on learner g_{m-1}
- The resulting classifier minimizes the exponential loss function $\sum_i \log(1 + \exp -y_i g(x_i))$.

AdaBoost Algorithm

- For m in $1, \ldots, M$
  - Set $\mathbf{p}^m = \frac{\mathbf{w}^m}{\sum_{i=1}^n w_i^m}$
  - For t in $1, \ldots, T$
  - 1 Fit base learner g_t and compute its error, $\epsilon_m = \sum_{i=1}^n p_i^m |g_m(x_i) - y_i|$
  - 2 If $\epsilon_m < \epsilon$
  - Set $\beta_m = \frac{\epsilon_m}{1-\epsilon_m}$
  - Update weights, $w_i^{m+1} = w_i^m \beta_m^{1-|g_m(x_i)-y_i|}$
  - 3 Else resample w_i^m and return to (1)
- $\hat{g}_f(x) = \left\{ \begin{array}{ll} 1, & \sum_{m=1}^M \log(\beta_m) h_m(x) \leq \frac{1}{2} \sum_{m=1}^M \log(\beta_m) \\ 0, & \text{otherwise} \end{array} \right\}$

AdaBoost in R

```
pacman::p_load(doParallel, adabag)
set.seed(333)
cl <- makeCluster(detectCores() - 1, port = 11999)
registerDoParallel(cl)
# Train and optimize tree, predict on test set
ada_tr = train(form = type ~ .,
               data = tr,
               trControl = ctrl,
               method = "ada")
stopCluster(cl)
saveRDS(object = ada_tr, file = "ada_train.rds")
pr_ada = predict(ada_tr,
                 newdata = te %>% select(-type))
```

AdaBoost Accuracy

- AdaBoost provides an improvement over classification trees
- On this data set, AdaBoost performs comparably to bagging but worse than random forest
- Accuracy = 95.74%

	Predictions	
True Response	nonspam	spam
nonspam	674	23
spam	26	427

Table 4. Confusion matrix for AdaBoost

Gradient Descent in Function Space: Gradient Boosting

- AdaBoost is limited by its use of the exponential loss
  - Sensitive to outliers and noise
- Gradient boosting algorithm proposed by Friedman (2001)
- Generalizes boosting to any loss function for which a gradient is well-defined
- Algorithm:
  - $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^n L(y_i, \rho)$
  - For $m \in 1, \ldots, M$
    - Update $\tilde{y}_i = -\left[\frac{\delta L(y_i, F(\mathbf{x}_i))}{\delta F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, \ldots, n$
    - $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^n [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
    - $\rho_m = \arg\min_\rho \sum_{i=1}^n L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
    - $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

Gradient Boosting in R

```
pacman::p_load(doParallel, xgboost)
set.seed(333)
cl <- makeCluster(detectCores() - 1, port = 11999)
registerDoParallel(cl)
# Train and optimize tree, predict on test set
gb_tr = train(form = type ~ .,
              data = tr,
              trControl = ctrl,
              method = "xgbLinear",
              allowParallel = TRUE)
stopCluster(cl)
saveRDS(object = gb_tr, file = "gb_train.rds")
pr_gb = predict(gb_tr,
                newdata = te %>% select(-type))
```

Gradient Boosting Accuracy

- Gradient boosting provides an improvement over classification trees
- On this data set, gradient boosting performs comparably to bagging and AdaBoost but worse than random forest
- Accuracy = 95.83%

	Predictions	
True Response	nonspam	spam
nonspam	673	24
spam	24	429

Table 5. Confusion matrix for gradient boosting

Cross Validation Estimation of Generalization Error

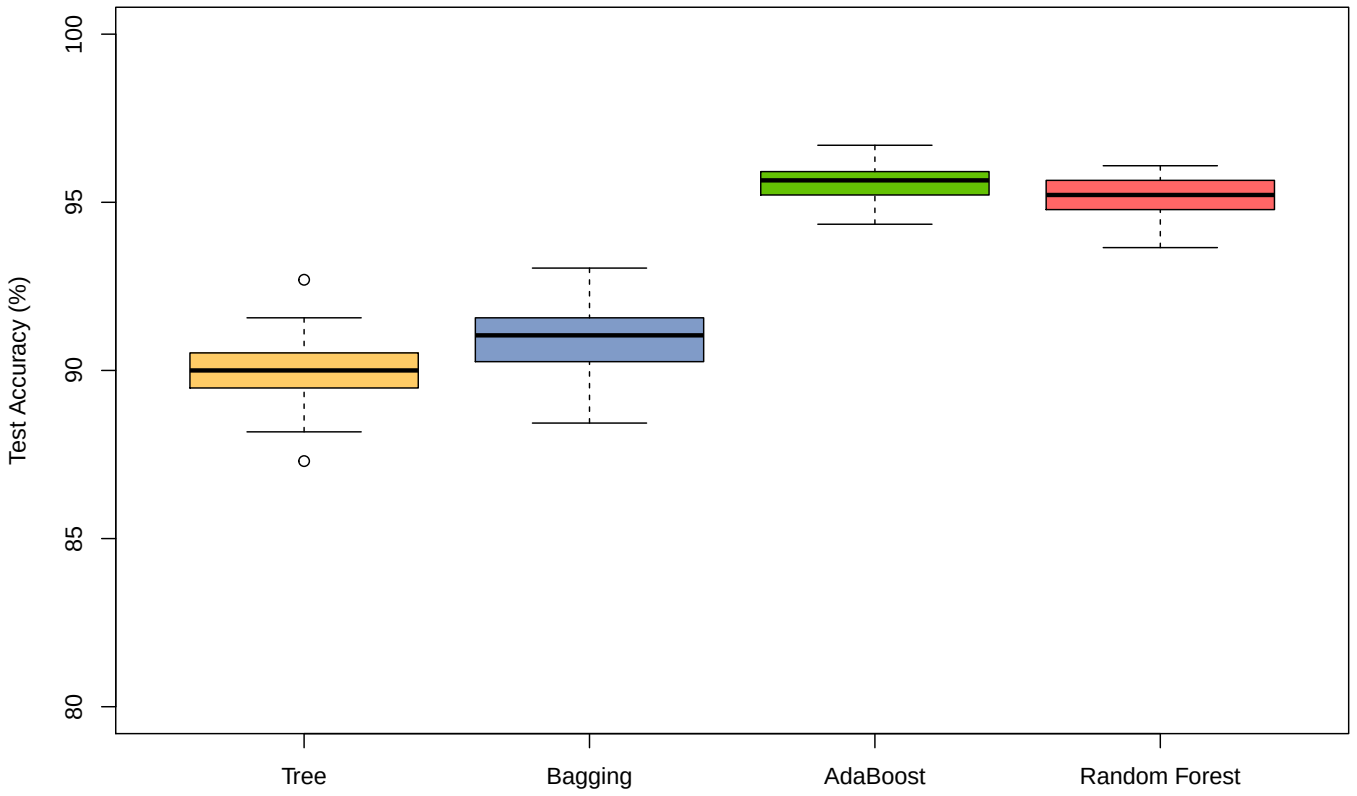

Figure 2. Box plots of the classification test error for the spam data set using 50 rounds of cross validation.

References

- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/ Cole Advanced Books & Software. ISBN 978-0-412-04841-8.
- Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123 - 140.
- Breiman, L. (2001). Random forests. *Machine learning, 45(1)*, 5-32.
- Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Computational Learning Theory*, 23-37.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189-1232.