Model Fitting

What we have been doing:

▸ get data
▸ fit some models
▸ evaluate results
▸ choose best model, apply to test data

What if we try something different?

▸ get data
▸ fit some models
▸ evaluate results
▸ combine models, apply to test data

This type of method is called an *ensemble*
Ensemble Methods

*Ensembles methods* use collections of models to get better predictive performance than any single model

- get a collection of predictive models
- the ensemble predictor is a weighted average of the underlying models

Why should this work?

- often easy to fit simple models well
- simple models have a limited hypothesis space
- if we average lots of *different* simple models, we can fit these well and have a large hypothesis space
- and we can reduce the variance of the estimator
Ensemble Methods

Let’s think about linear classifiers: divide labels with a hyperplane.

None divides the data particularly well, but better than random guessing... how can we combine them?
Ensemble Methods

Let's think about linear classifiers: divide labels with a hyperplane.

Combine by assigning majority labels... now non-linear!
Trees:

- trees are really flexible—good for regression and classification
- they tend to fit pretty well, but often do not have the best predictive error
- they are also unstable
Ensemble Methods and Trees

Instability:
- small changes in data (or fitting method) produce big changes in outcome
- this is good for ensembles! diverse results!

Idea: randomize the data (or tree building method) a bit to get even more diverse results!
Bagging

Idea: to get a set of trees, randomize the data
- use bootstrapping to get new datasets
- in bootstrapping, we create a new dataset by sampling with replacement from our current dataset
Bagging

Idea: to get a set of trees, randomize the data

▶ use bootstrapping to get new datasets
▶ in bootstrapping, we create a new dataset by sampling with replacement from our current dataset
▶ do this $K$ times
▶ fit a tree to every new dataset
▶ new estimator is average of bootstrap estimators
▶ called Bagging, or Bootstrap aggregating
Example: Kyphosis data

# Make a bagged estimate for kyphosis

K <- 100
n.train <- 60
n.val <- length(kyphosis$Kyphosis)
n.test <- n.val - n.train
inds.train <- sample(1:n.val,n.train)
inds.test <- setdiff(1:n.val,inds.train)

kyphosis.train <- kyphosis[inds.train,]
kyphosis.test <- kyphosis[inds.test,]
Example: Kyphosis data

# Make a bagged estimate for kyphosis

```r
out.list <- list(K)
out.vals <- mat.or.vec(n.test,K)

base.fit <- rpart(Kyphosis ~ Age + Number + Start,
    data=kyphosis.train,parms=list(split='information'))
out.vals.base <- predict(base.fit,kyphosis.test,type="class")
```
Bagging

Example: Kyphosis data

# Make a bagged estimate for kyphosis

for (i in 1:K){
    # Make a new training set
    inds <- sample(1:n.train,n.train,replace=TRUE)
    df.temp <- kyphosis.train[inds,]
    fit.temp <- rpart(Kyphosis ~ .,
        data=df.temp,parms=list(split='information'))
    out.list[[i]] <- fit.temp
    out.vals[,] <- predict(fit.temp,kyphosis.test,type="class")
}

bag.pred <- mat.or.vec(n.test,1)
for (i in 1:n.test){
    n.temp <- mean(out.vals[i,])
    if (round(n.temp)==2) bag.pred[i] <- "present"
    else bag.pred[i] <- "absent"
}
Bagging

Example: Kyphosis data

\[ \text{mean}((\text{out.vals.base}!=\text{kyphosis.test}\$\text{Kyphosis})) \]
\[ \text{mean}((\text{bag.pred}!=\text{kyphosis.test}\$\text{Kyphosis})) \]
\[ \text{par(mfrow=c(1,4))} \]
\[ \text{plot(out.list[[1]])} \]
\[ \text{text(out.list[[1]],use.n=FALSE)} \]
\[ \text{plot(out.list[[2]])} \]
\[ \text{text(out.list[[2]],use.n=FALSE)} \]
\[ \text{plot(out.list[[3]])} \]
\[ \text{text(out.list[[3]],use.n=FALSE)} \]
\[ \text{plot(out.list[[4]])} \]
\[ \text{text(out.list[[4]],use.n=FALSE)\]
Example: Kyphosis data

- Start $\geq 8.5$
  - sent
  - present
- Start $\geq 14.5$
  - Age $< 36$
    - absent
    - present
  - Start $\geq 10.5$
    - absent
    - present
- Start $\geq 12.5$
  - Number $< 4.5$
    - absent
    - present
- Start $\geq 14.5$
  - Age $< 44$
    - absent
    - present
  - Start $\geq 10.5$
Bagging

Why does bagging work?

Recall:

\[ MSE = (Estimator\ bias)^2 + (Estimator\ variance) \]

- bias is about the same between estimators
- averaging over (diverse!) estimators reduces variance
- ...which means lower predictive error
Bagging

Pros:
▶ easy to implement
▶ works better than trees on their own
▶ very fast and parallelizable

Cons:
▶ tends not to work as well as boosting (talk about this on Thursday)
▶ works best with high variance, low bias, low correlation estimators
Ensembles tend to work best with “not too complicated” estimators. Why?
Ensembles tend to work best with “not too complicated” estimators. Why?

- simpler estimators are often less correlated
- may cover a larger part of the hypothesis space

Can we get these characteristics with bagged trees?
Random Forests

Want to *decorrelate* bagged trees. What if we grow trees randomly?

- follow same branching path if inputs are the same
- small changes in inputs can lead to large changes in outputs
- so what if we force the trees to split on different attributes?

Randomly select a subset of attributes that it can split on!
Random Forests

Idea: another way to get diverse estimators is by randomizing the tree generation

- randomly select a small subset of features before each split
- fit a tree on the reduced data
- don’t prune!
- do this $K$ times
- average results to get new predictor
- called random forests
Random Forests

Why does this work?

- suppose we have $K$ random variables, $X_1, \ldots, X_K$
- each has variance $\sigma^2$
- pairwise correlation $\rho$
- set $Y = \frac{1}{K} \sum_{k} X_k$

Then,

\[
\mathbb{E}[Y] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[X_k]
\]

\[
\text{Var}(Y) = \rho \sigma^2 + \frac{1 - \rho}{K} \sigma^2
\]
Random Forests

Practical matters: How many features? What is tree depth?

Purely heuristic:

- **Classification**: $m = \text{floor}(p)$ and minimum node size is 1
- **Regression**: $m = \text{floor}(p/3)$ and minimum node size is 5
Random Forests

Variable importance:

- how important is each variable to the model?
- at each tree split, the improvement in the split criterion is the importance given to the splitting variable
- add importance over all trees

**FIGURE 15.5.** Variable importance plots for a classification random forest grown on the spam data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 354). The right plot uses OOB randomization to compute variable importances, and tends to spread the importances more uniformly.
To implement random forests in R, use `randomForest` package

- it works very similarly to `rpart`

```r
> library(randomForest)
> tree.rf <- randomForest(Kyphosis ~ Age + Number + Start,
                         data=kyphosis.train)
> out.rf <- predict(tree.rf, kyphosis.test)
```
Ensembles and Trees

- Ensembles are a great way to deal with the stability issues of trees
- Usually produce results with better predictive error
- Random forests are a great black box method for classification and regression—often a good first (or second or third) method to try
- Scales well to large datasets ($\mathcal{O}(10^7)$ observations, thousands of covariates)
- However, results are not as interpretable as individual trees
Applications of Random Forests

http://www.youtube.com/watch?v=HNkbG3KsY84
Applications of Random Forests

depth image $\rightarrow$ body parts $\rightarrow$ 3D joint proposals
Applications of Random Forests

- start with about 500k labeled poses from motion capture
- randomly generate millions more over different poses and body types
Applications of Random Forests

To make a classification tree for an image, first create a set of features:

\[ f = d(x + u/d(x)) - d(x + v/d(x)) \]

- \((u, v)\) is a set of displacement vectors
- \(d(x)\) is the distance from the Kinect at pixel \(x\)
- not a natural metric, but measures some part of the 3D distance around each pixel
Applications of Random Forests

- create a set of features
- for each image, select about 2,000 pixels
- create tree classifier for this image using features
- use random forest to create distribution over body parts for new image
- new images can be classified in about 2ms on XBox 360 GPU
Homework 5 is now posted. Let’s go through this.
The midterm grades were a lot lower than expected (about 20 point drop from spring).

Grades are done by points. Current breakdown is 40% homework, 25% midterm, 35% final. Previously, distributions have been as follows:

- overall percentage is high 80’s and above: some form of A
- overall percentage low to mid 70’s to mid 80’s: some form of B
- below that: instructor discretion

That roughly means on tests 80-100 is A range, 60-80 is B range, 40-60 is C range, below 40 is problematic.
The midterm grades were a lot lower than expected (about 20 point drop from spring).

Proposal:

- you have the option to change midterm/final weights from 25/35 to 15/45
- a question will be added to Courseworks for your selection (no selection means 25/35)
- you need to make a selection before the last day of class
- the final will be harder than the midterm!