第13章　ツリーモデル：補足

木から森へ：アンサンブル学習

ツリーモデルは樹形図で表示することができるため，その結果を人々に容易に説明できるという利点を持ちます．しかし，この説明可能性の利点と裏腹に，予測性能が低いという欠点を持ちます．複数のツリーを融合させる**アンサンブル学習**を用いることによって，この欠点を改善できます．ここではアンサンブル学習の代表的な手法であるバギングとランダムフォレストを適用して 13.2 節で構築した回帰木の予測性能の向上を試みます．

S.1 バギング

説明変数 $\boldsymbol{x} = (x_1, \ldots, x_p)$ に対する y の予測分散を小さくしたいとします．いま仮に $(\boldsymbol{x}, y)$ に関する大きさ n の独立な標本が B 組あるとします．その b 番目の標本

$$\{(\boldsymbol{x}_{1b}, y_{1b}), (\boldsymbol{x}_{2b}, y_{2b}), \ldots, (\boldsymbol{x}_{nb}, y_{nb})\}$$

から回帰木（または決定木）を構築し，$\boldsymbol{x}$ に対する予測値 $\hat{y}_b(\boldsymbol{x})$ を求めます．このようにして求めた B 個の予測値の平均 [1)]

$$\hat{y}_{\text{avg}} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b(\boldsymbol{x})$$

の分散の大きさは，オリジナルなデータに対する予測値の分散の B 分の 1 に低下します．

[1)] 分類木の場合は，平均の代わりに最頻値を使って集計します．

実際には観測データは一組しかありません．**バギング**（bagging）[2)]では，ブートストラップ法という再標本抽出法によって標本を B 組発生します．ブートストラップ法では，オリジナルなデータ $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ を母集団とみなして，そこから大きさ n の標本を無作為に復元抽出します．

S.2 ランダムフォレスト

ランダムフォレスト（random forest）は，バギングと同様に，ブートストラップ標本を用います．ただし，p 個すべての説明変数を用いるのでなく，それらから無作為抽出した m 個の説明変数を用います．しばしば $m = \sqrt{p}$ と設定します．説明変数も無作為標本の対象とする理由は，分散減少の効果を確保するためです．たとえば，いま非常に予測力の高い説明変数があったとします．この場合，ほとんどのブートストラップ標本では，この変数を用いて最初に分割するでしょう．そのため，多くのツリーが似通ったものになってしまいます．その結果，ツリー同士の相関が高くなってしまい，期待していたような分散減少効果を得られなくなります．

バギングやランダムフォレストは予測誤差を減少させますが，その一方で結果の解釈性も低下します．しかし，変数間の影響度の相違を重要度（variable importance measure）によって判断できます．ある変数の重要度は，その変数を用いた分割による RSS（あるいはジニ係数）の減少の総和を B 個のツリーについて平均した値です．

S.3 マンション取引データへの応用

randomForest という R パッケージを用いて，13.1 節で取り上げたマンション価格の回帰木のバギングとランダムフォレストを行いましょう．まず，13.1 節と同様にして訓練データとテストデータを作成します．

```
> condo <- read.csv("condo.csv")
> condo_res <-subset(condo, condo$zoning=="住居")
> set.seed(123)
> id_test <- sample(1:nrow(condo_res), floor(nrow(condo_res)/
  2)) # テストデータの個数
> test_data <- condo_res[id_test,] # テストデータの作成
> train_data <- condo_res[-id_test,] # 訓練データの作成
```

[2)] **b**ootstrap **agg**regation の略語です．

まず，バギングを実行します．

```
> install.packages("randomForest")
> library(randomForest)
> set.seed(123)
> bag_condo <- randomForest(price~., train_data, mtry=5,
  ntree=1000, importance=TRUE)
> importance(bag_condo, type=1)  # 重要度の表示
```

```
                %IncMSE
fspace     102.642004029
age         64.319653514
distance     3.321527923
structure   -0.003398485
zoning       0.000000000
```

```
> yhat_bag <- predict(bag_condo, newdata=test_data)
> (rmse_bag <- sqrt(mean((yhat_bag-test_data$price)^2)))
```

```
[1] 9.280745
```

ここで，mtry は説明変数の個数です．mtry=5 としてすべての説明変数を用いると，ランダムフォレストはバギングに一致します．平方根平均二乗誤差は小さくなりました．fspace の重要度が一番大きく，その後に age の重要度が続きます．

続いてランダムフォレストを実行します．

```
> set.seed(123)
> for_condo <- randomForest(price~.-id, data=train_data,
  mtry=3, ntree=1000, importance=TRUE)
> importance(for_condo, type=1)
```

```
             %IncMSE
fspace     88.003096
age        49.489635
distance    2.806871
structure   1.228885
zoning      0.000000
```

```
> yhat_for <- predict(for_condo, newdata=test_data)
> (rmse_for <- sqrt(mean((yhat_for-test_data$price)^2)))
```

```
[1] 9.762486
```

通常の回帰木に比べ平方根平均二乗誤差が改善されました．ただし，その改善はバギングに劣ります．