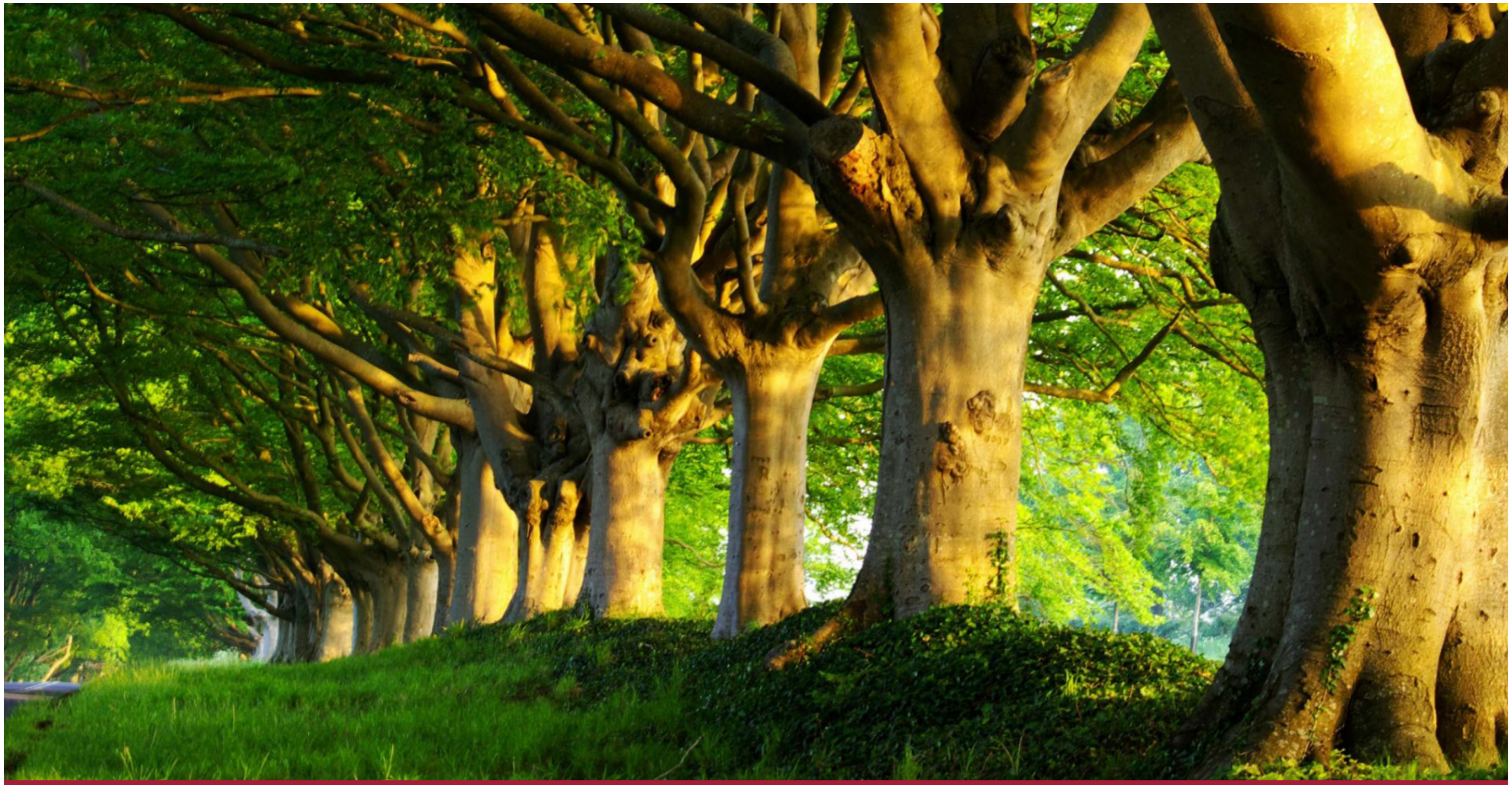
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Philosophische Fakultät
Seminar für Sprachwissenschaft



Gradient boosting machines

Peter Hendrix

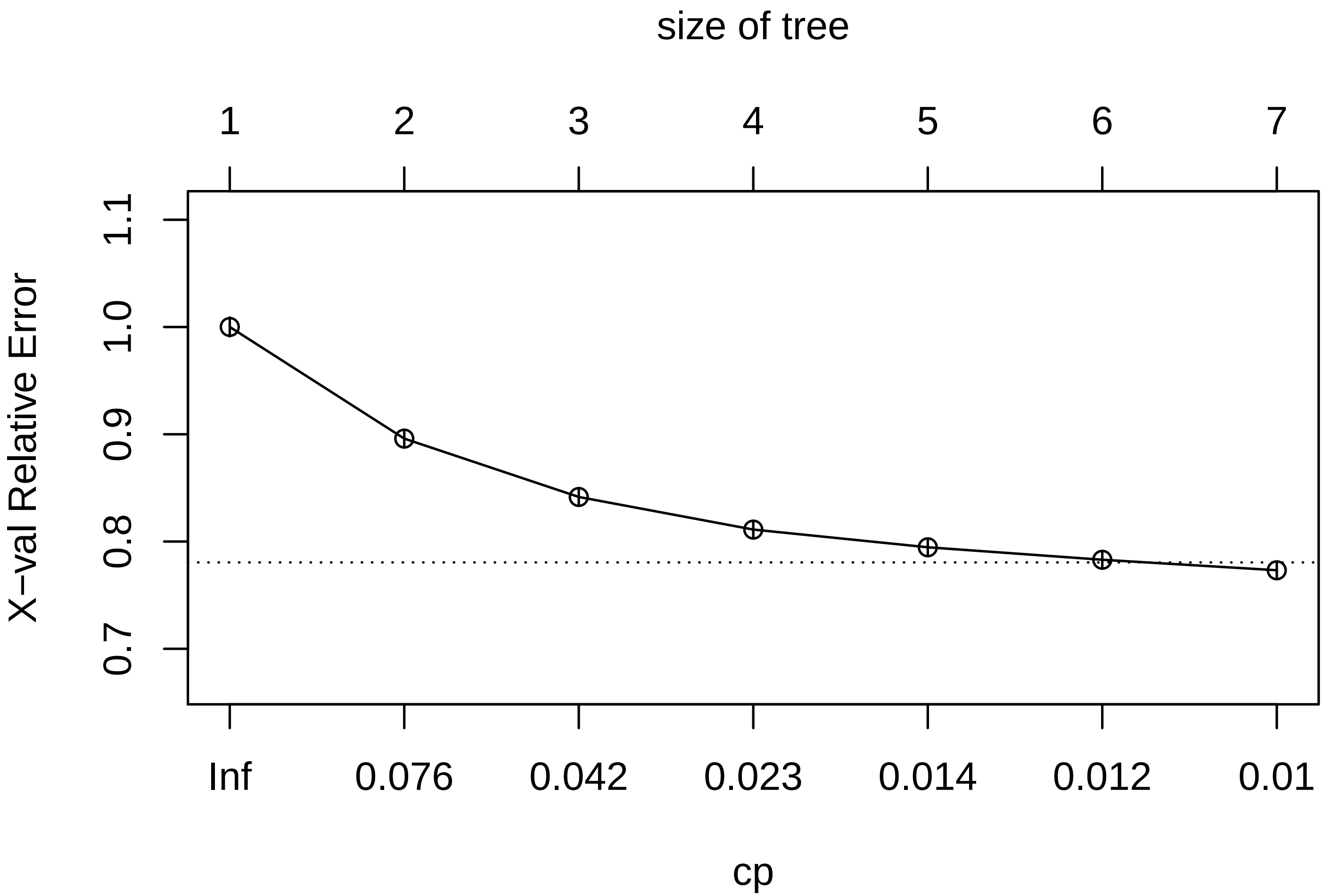
Regression trees

```r
# Load Chinese naming data
load("data/datachinesenaming.rda")

nrow(data)

# [1] 30665

ncol(data)

# [1] 76
```

Regression trees

```
library(rpart)
rpart = rpart(RTinv ~ ..., data = data)

plotcp(rpart)
```

Regression trees



Regression trees

```
rpartpruned = prune(rpart, cp = 0.012)

cor(datatmp$RTinv, predict(rpartpruned))^2
# 0.2204487
```


Regression trees

```
controls = ctree_control(maxdepth = 20)
ctree = ctree(RTinv ~ ..., data = data,
              controls=controls)

cor(datatmp$RTinv, predict(ctree))^2
# 0.3482986
```

Ensembles

Why plant a single tree if you can grow a forest?

Ensemble methods

- Examples:
  - bagging
  - random forest
  - boosting

Bagging

- Bootstrap aggregating
- Build large number of trees on samples of the data

Random forest

- Consider only a random subset of N predictors out of all predictors P for each split
- $N = \sqrt{P}$ tends to work well
- Random forests are identical to bagging if N is equal to P

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Random forest

```
library(party)

controls = cforest_unbiased(ntree = 100,
                            mtry = round(sqrt(69)))

forest = cforest(RTinv ~ ..., data = data,
                 controls = controls)

cor(data$RTinv, predict(forest))^2
# 0.6052593
```

Gradient Boosting Machine

- Trees are grown sequentially
- Each tree is an expert on the errors of its predecessor

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Gradient Boosting Machine: step 1

- Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the data

Gradient Boosting Machine: step 2

- Repeat for $b = 1, 2, ..., B$:
  - Fit a tree $\hat{f}^b$ to the residuals
  - Update $\hat{f}$ by adding a shrunken version of the new tree:
    $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
  - Update the residuals:
    $$r_i \leftarrow r_i - \lambda \hat{f}^b(x)$$

Gradient Boosting Machine: step 3

- Output the gbm model: $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Gradient Boosting Machine

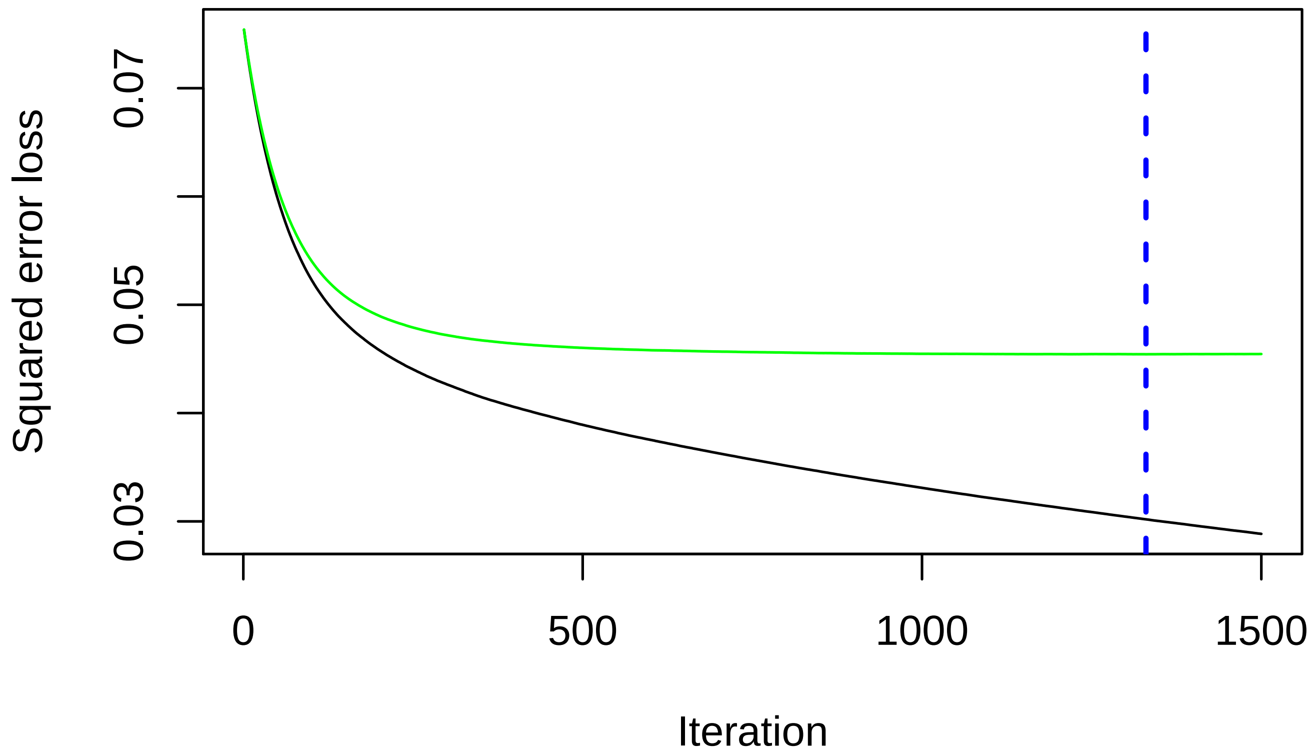
- Parameters:
  - Number of trees
  - Shrinkage (λ)
  - Number of splits in the tree (interaction depth)

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Gradient boosting machine

```r
library(gbm)
gbm = gbm(RTinv ~ ..., data = data,
          interaction.depth = 40, shrinkage = 0.01,
          n.trees = 1500, cv.folds = 10, n.cores = 10)

gbm.perf(gbm)
# Using cv method...
# 1330
```

Gradient boosting machine



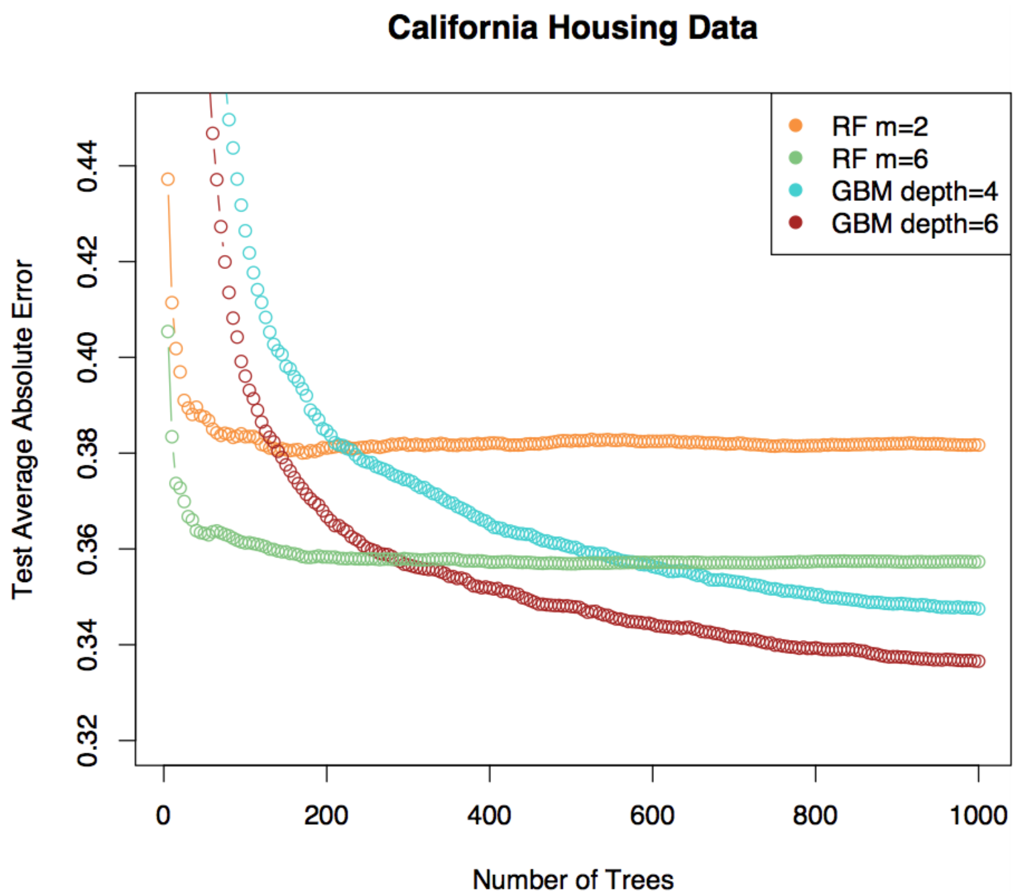
Gradient boosting machine

```
data$Predict = predict(gbm, n.trees = 1330)
cor(data$Predict, data$RTinv)^2
# 0.6193783
```

Gradient boosting machine

```
summary(gbm)


                          rel.inf
InitialPhoneme        20.675422635
LogChar1FamFreqZ       7.848921035
LogChar1FreqZ          7.674469913
Session                6.187537938
LogFrequencyZ          5.303294727
FinalPhoneme           3.253571436
LogChar1FriendsZ       2.834133767
LogChar1StrokesZ       2.464799656
...                            ...
Total                          100
```

Gradient boosting machine



Gradient Boosting Machine

- Extremely competitive performance
- Overfitting not much of an issue
- Interpretation more limited than for regression models
- Computationally expensive due to sequential fitting