Random Forests™

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

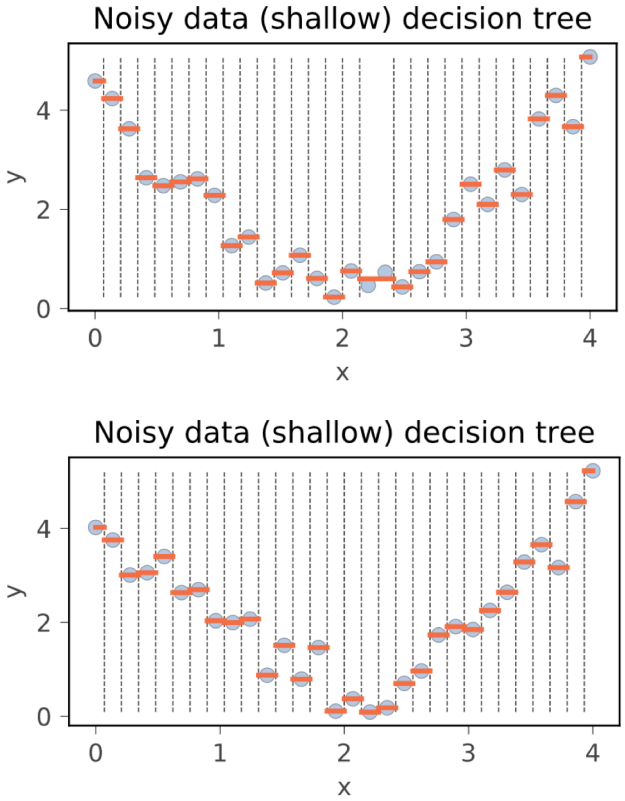
RF motivation

Leo Breiman (1996) introduced bagging
then Random Forests (2001)
https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

- Decision trees can often get training errors close to zero because we can grow very large trees to partition the feature space into tiny regions with 1 or just a few observations; trees have very **low bias**
- The downside is that decision trees overfit or, using stats nerds terminology, decision trees have **high variance** and don't generalize well
- Goal: keep the low bias, but reduce the variance (increase the generality)

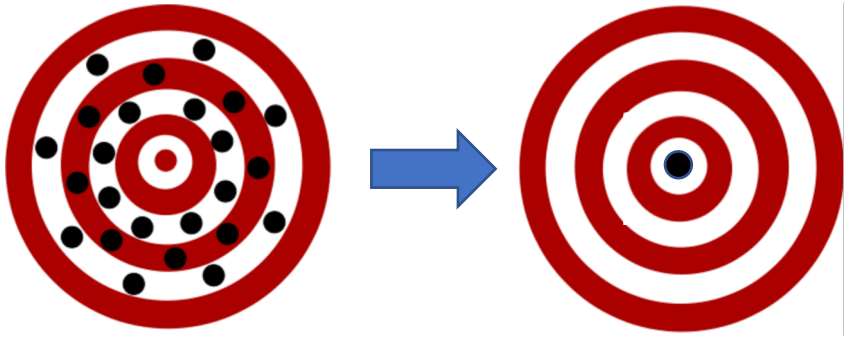
Variance is the key word here

- Decision trees are sensitive to quirks in the data and so similar but different training sets drawn from the same distribution can yield very different models (parameters)
- Same fit alg. on two i.i.d. X training sets:
- The predictions made by decision trees trained on i.i.d. sets bounce around, but their average approaches the correct value (same for i.d. sets)



Ensemble of high-variance decision trees

- Decision trees are inconsistent but are accurate on average across multiple same distribution (i.d.) training sets
- Pretend we do have multiple i.d. X and, hence, multiple models
- Then, we can average predictions from multiple models to get an accurate prediction:



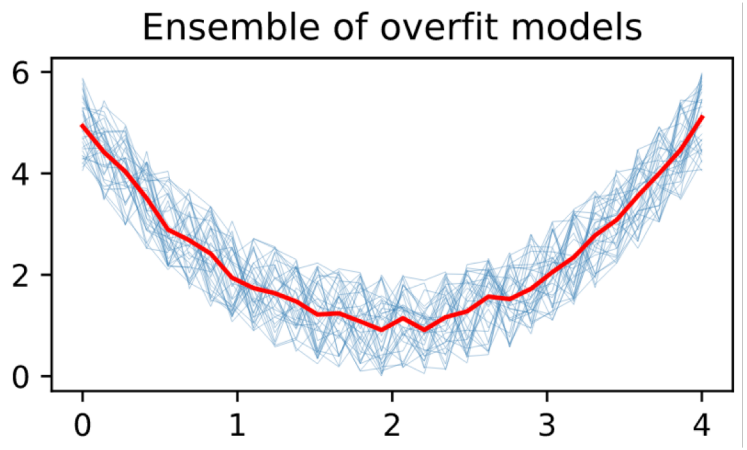
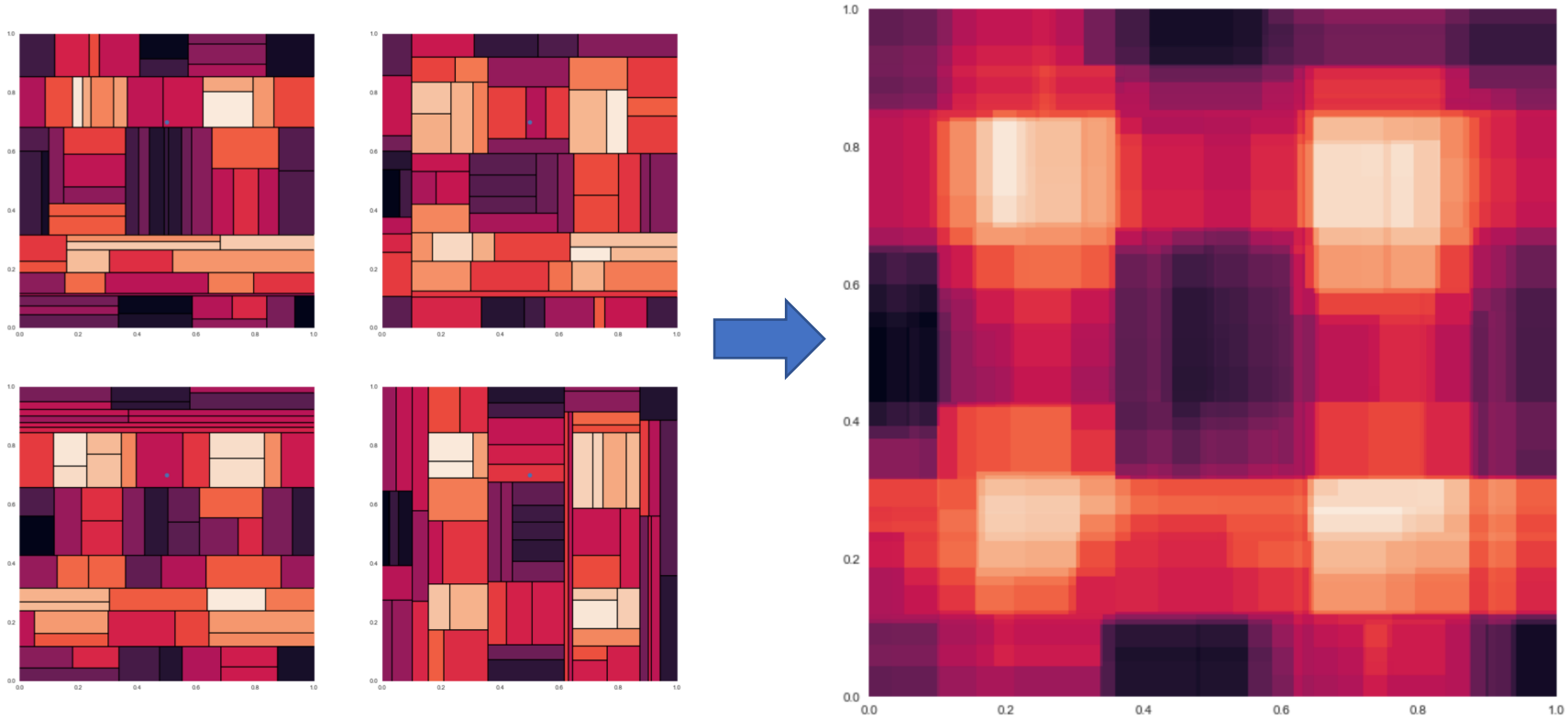


Image credit: http://ficscience.blogspot.com/2011/02/technicalities-of-technical-terms.html

RF: Overlapping feature space partitions



Images generated by USF MSDS alumnus Tyler White
https://gist.github.com/tylerwx51/fc8b316337833c877785222d463a45b0

Ensemble's effect on bias and variance

- If each X is identically distributed (doesn't have to be i.i.d.), the average of the tree predictions is the same as the expected prediction from any tree trained on one of the X; E[X_i] = μ
- That's good because trees have low bias
- Central limit theorem says that if variance of *T* i.i.d. random variables is σ^2, the variance of the average is $\frac{\sigma^2}{T}$
- So, as we add trees, the variance of the prediction (average from ensemble trees) tightens!

Bagging (Bootstrap aggregation)

- We only have one training set, X, but can bootstrap to get as many as we want from the same distribution (these are i.d.)
- *Bootstrap*: from n records, randomly select n with replacement
- Each bootstrapped data set will have about 63% of the unique overall records
- *Aggregation*: average the ensemble predictions (or majority vote for classification)
- Analogy: real estate agents sent out to sample NYC apartments/prices; there will be some overlap but average of all agents' predictions is more accurate than an individual's

show diff trees

Problem is bootstrapping is i.d. not i.i.d.

- If just identically distributed and not necessarily independent, with positive pairwise correlation ρ between trees, then variance of the average for random vars is: (Eqn 15.1 ESLII Hastie *et al*):

$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

- As num trees, T, grows, 2nd term disappears but correlation of trees would still have variance proportional to ρ (at most σ^2)
- Bagging with ρ>0 helps to reduce variance but can do better

What does “correlated trees” mean?

- Model parameters are correlated; “not independent” better term
- **Analogy**. Real estate agents sampling bias: if they visit apartments they observe other agents visit, variance might not shrink with more agents; worst-case: they all visit the same apartments; best case is they choose all apts independently
- **Worst-case**. Imagine replicating single decision tree T times, variance of average prediction would be same as variance of single tree (i.e., bad strategy); if ρ=1, variance of avg = σ^2
- **Best-case.** if ρ=0, trees are uncorrelated and variance of average converges to 0 as T grows

- Model parameters are correlated; “not independent” better term
- **Analogy**. Real estate agents sampling bias: if they visit apartments they observe other agents visit, variance might not shrink with more agents; worst-case: they all visit the same apartments; best case is they choose all apts independently
- **Worst-case**. Imagine replicating single decision tree T times, variance of average prediction would be same as variance of single tree (i.e., bad strategy); if ρ=1, variance of avg = σ^2
- **Best-case.** if ρ=0, trees are uncorrelated and variance of average converges to 0 as T grows

Bagging summary

- Bagging does not increase **bias**; the avg of tree predictions is same as the expected prediction from any 1 bootstrapped tree
- Keep in mind, however, that a tree fit to bootstrapped data is only using 63% of the data and so each RF tree will have higher bias than single decision tree fit to entire data set
- Bagging does reduce **variance**, it is averaging predictions from lots of non-identical models afterall
- Degree of tree similarity influences ability to reduce variance to 0
- If ρ=1, no reduction in variance, but ρ=0 means adding trees reduces variance to 0 asymptotically

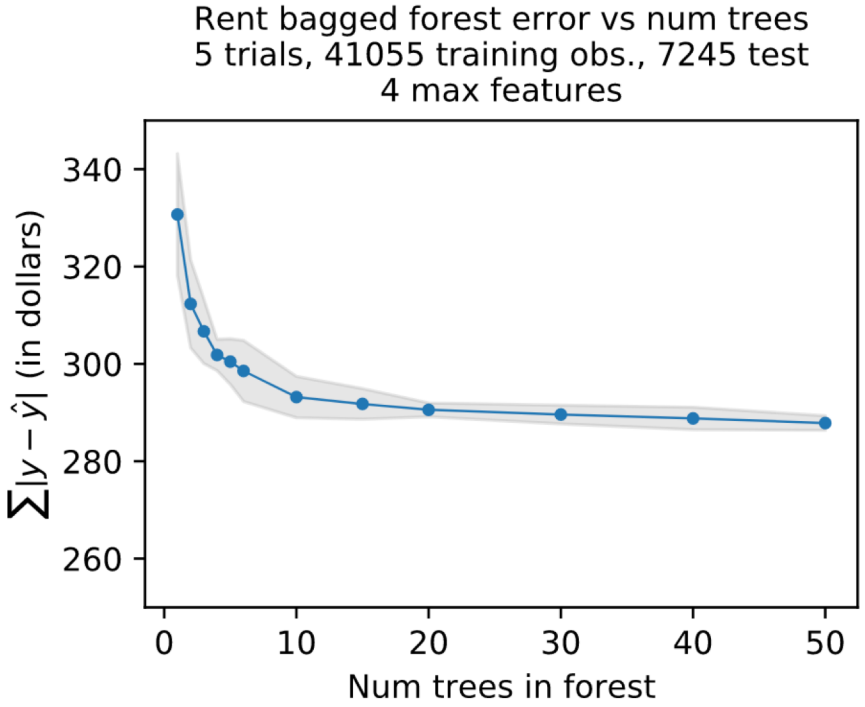
De-correlating trees

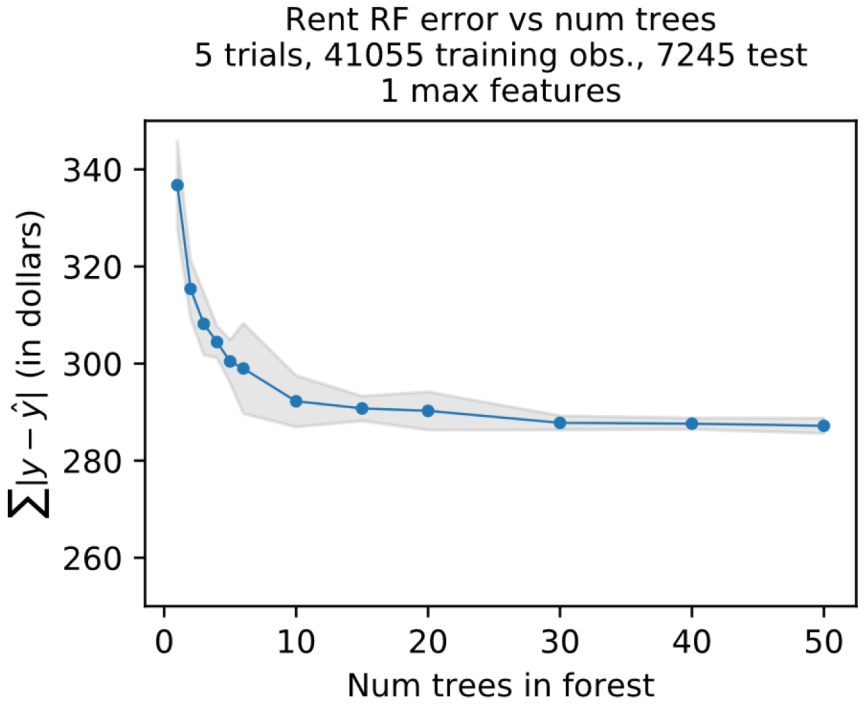
$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

- Bagging overcomes most of the overfitting (Hastie *et al* p589 say ρ usually small like 0.05)
- Can do a little better by de-correlating the trees to reduce ρ by restricting features available as splitting candidates per split
- Imagine one strongly predictive var out of p, then all trees will be similar; initial root splits, and many others, will likely be same
- Choosing max features, m < p: must make sure chance of selecting predictive variables is high (See ESLII p596)
- Let validation error be your guide

Compare bagged trees vs RF (Rent)

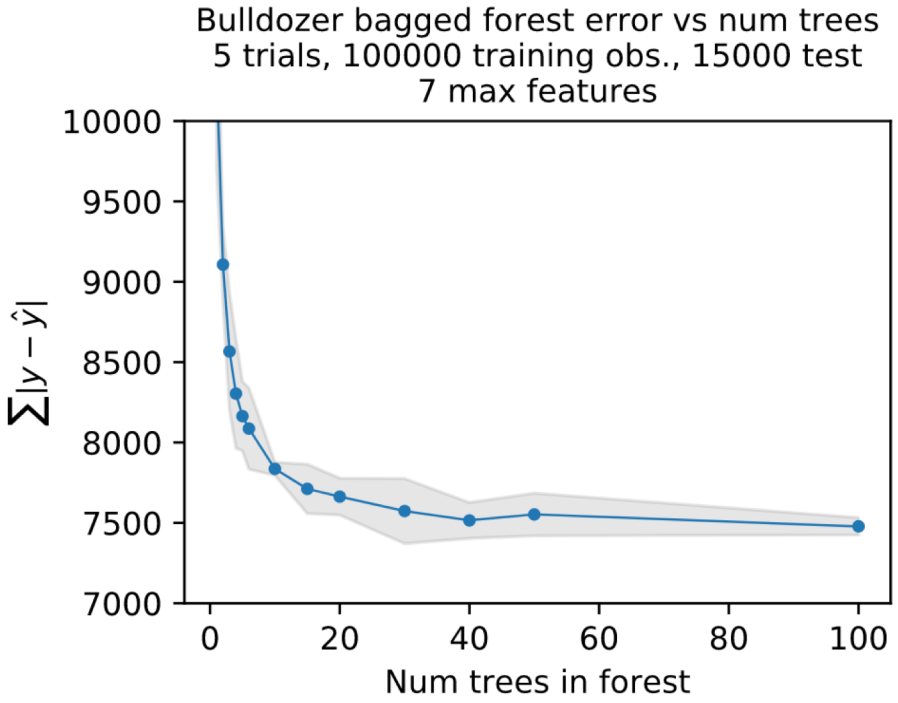
Very little difference with just p=4 features

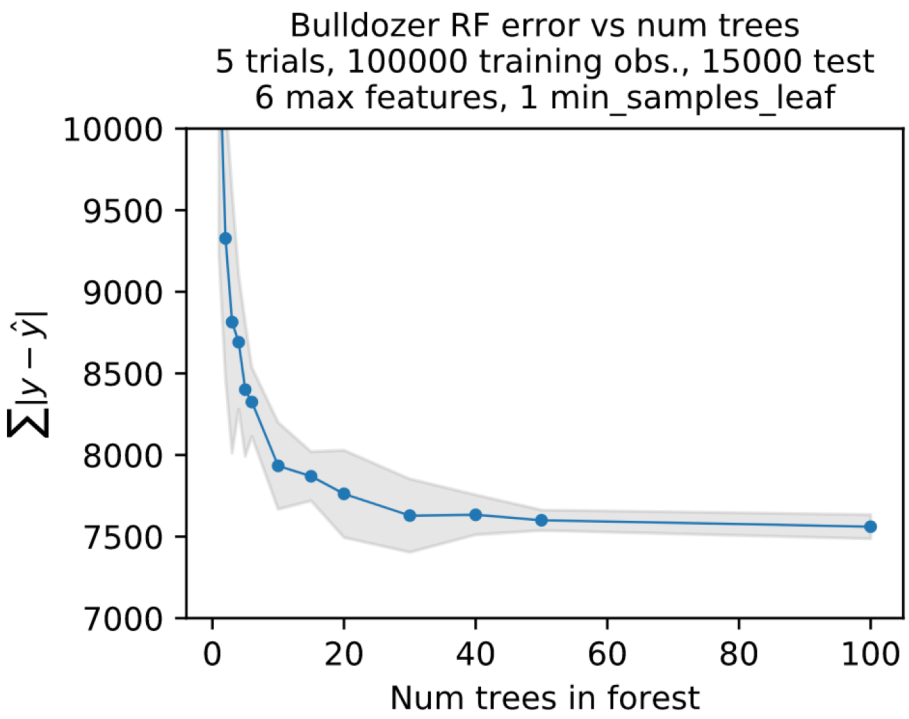




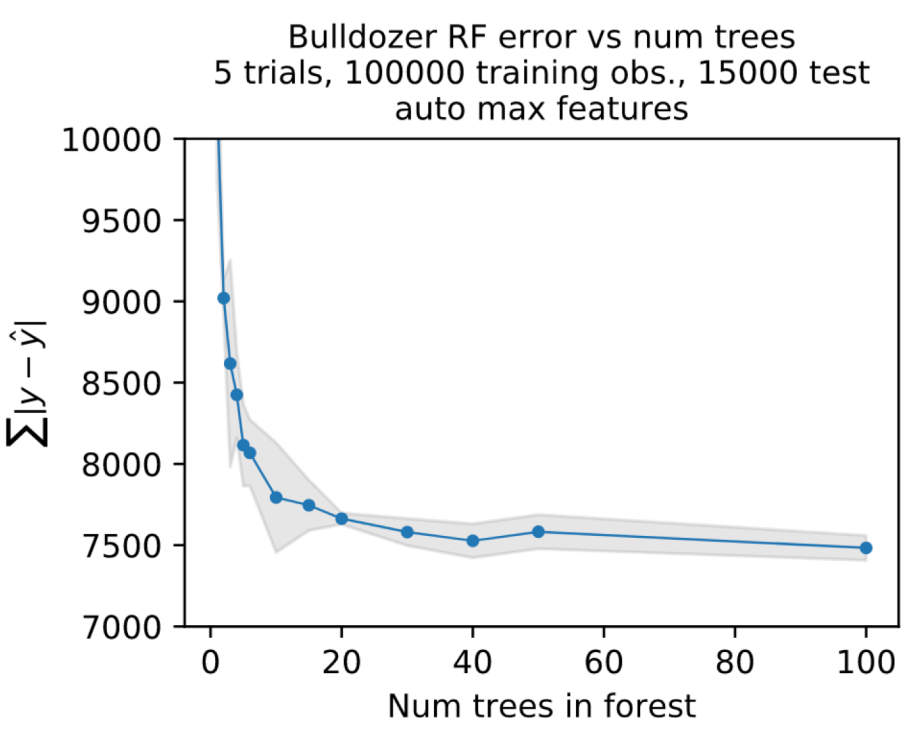
Compare bagged trees vs RF (bulldozer)

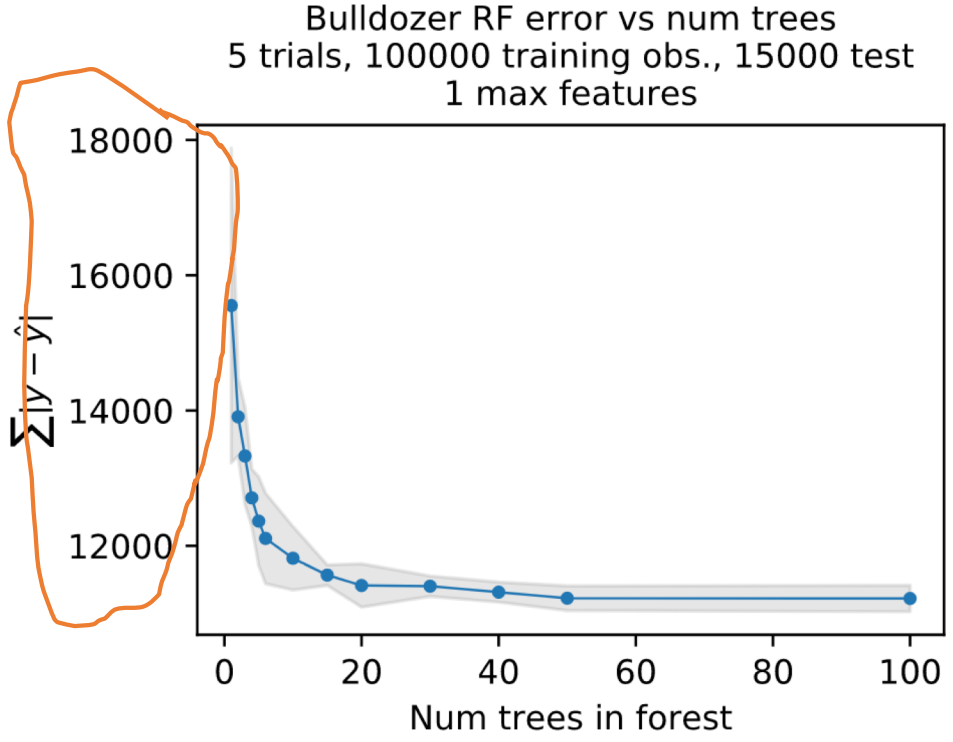
For this data, using all p=7 is best





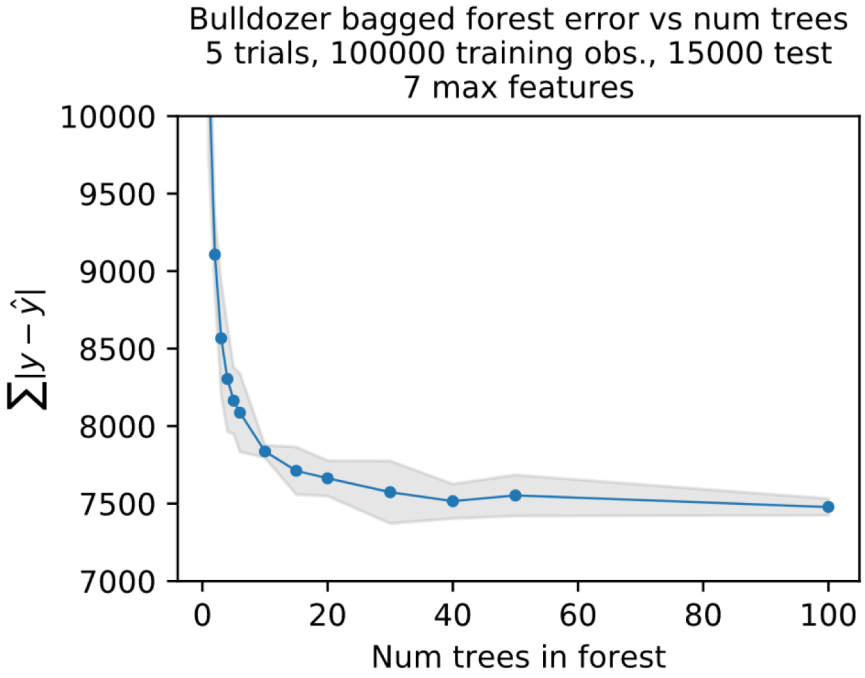
If max_features too low, high bias





Effect of forest size on bias

- Why does accuracy improve greatly (initially) as we add trees?
  - Each tree sees only 63% so adding bootstrapped trees increases use of training data
- Why does accuracy asymptotically approach a minimum instead of continual improvement?
  - With enough trees, ensemble sees 100% of the training data; it's approaching the bias of single decision tree in ideal world where it's not overfit



Properties (Breiman 2001)

- p4 "*Random forests do not overfit as more trees are added*" **Why**?
  - Adding more trees actually REDUCING variance and overfitting==variance
  - New trees get averaged in so each new tree has less individual effect
- p7 "*It's relatively robust to outliers and noise*" **Why**?
  - Outliers get shunted to their own leaf since doing so reducing loss function, particularly if squared-error is used; noise vars are predictive so not chosen as split vars
- p10 Bagging helps more the more unstable the model. **Why**?
  - Averaging is a smoothing operator, squeezing predictions to centroid
  - If model is low variance already, there is no point in bagging

Properties continued

- RFs are scale and range insensitive in features and target y
  - Comparing feature values in decision nodes not doing math on them
  - Computing mean or mode of y to predict
- ESLII p596 "*Classifiers are less sensitive to variance [than regressors]*"
  - (not sure why haha)

Bootstrapping vs subsampling

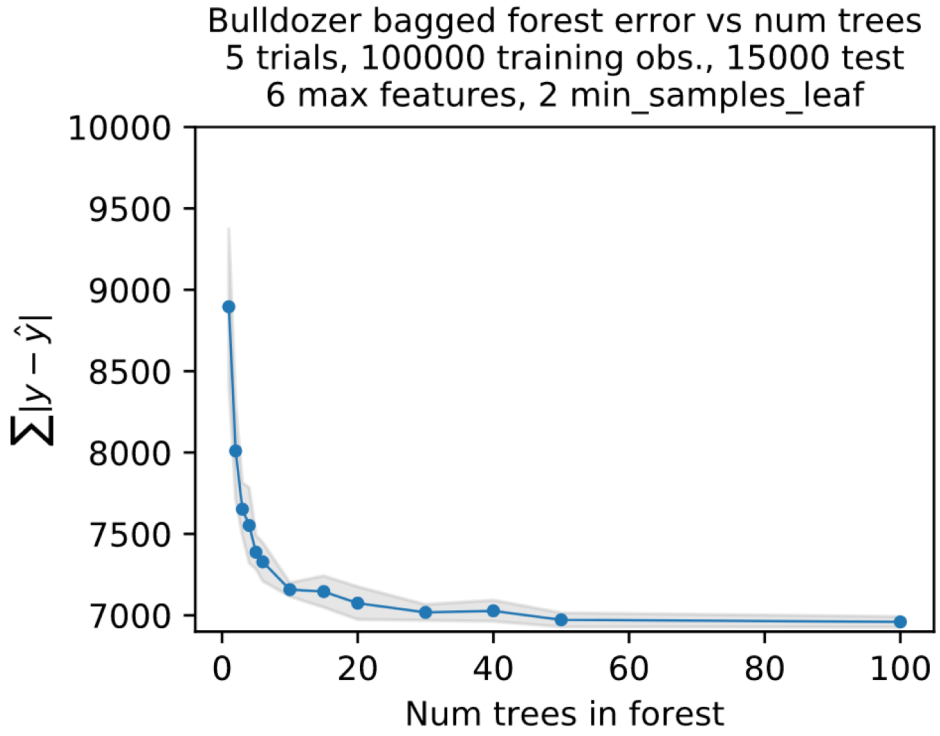
- Bootstrapping is sampling with replacement vs subsampling w/o replacement
- Friedman and Hall (2000): subsampling also works, showing that training trees with n/2 subsamples is similar in bias/variance to bagging http://statweb.stanford.edu/~jhf/ftp/bag.pdf
- Smaller training set is a big win in terms of speed
- Using even smaller fractions of n improve generality (reduce variance) because trees are less correlated (they work on different data chunks)

RF Tuning strategy

- Good news: very little tuning needed
- Start with 30 trees and work upwards til validation error stops getting better; or just pick 100
- Sklearn uses max_features=1.0 (all features) by default; try dropping this using sqrt(p), log(p), or similar; ESLII suggests p/3 for regression and sqrt(p) for classification
- Try adjusting min samples per leaf: 1, 3, 5, 10, 25, 100
- Goal: min validation error
- Can also try grid search, but I never bother

Feature engineering beats model tuning

- SalesID: unique record ID, and is never seen again
- Is that useful for prediction?
- Does the model think it's useful? Yes
- Model is overfit not on noise but on falsely-predictive feature
  - Could be that sales ID correlates with inflation or change in type of models sold in auction creates "trend" in sale prices
- A case where using LESS data improves the model a lot ($500 diff)
- Dropping useless features also often gives a small bump



Extremely randomized trees (Geurts *et al* 2006)

- The var/value pair is highly sensitive to the training set, and responsible much of error rate

  https://link.springer.com/article/10.1007/s10994-006-6226-1
- Geurts wondered if more randomness could reduce variance futher
- Pick random split value in min(X[:,j]) .. max(X[:,j]), ignoring y!
- Like RF, select m variables and choose var/value with lowest loss
- Fits using entire X training set, not bootstrap and not subsample (trying to reduce bias)
- Our use of 11 X candidate values in the project is similar (an effort to reduce variance and increase speed)

The RF algorithms

UNIVERSITY OF SAN FRANCISCO

Fitting RFs

Algorithm: $RFfit(X,\ y,\ loss,\ ntrees,\ max\_features,\ min\_samples\_leaf)$

for $i = 1..ntrees$ **do**

$X', y' = bootstrap(X, y, size = |X|)$

$T_i = RFdtreefit(X', y', loss, max\_features, min\_samples\_leaf)$

end

For regression, pass in loss of MSE or stddev
For classifier, pass in loss of gini

Fitting a single tree in RF

```
1 Algorithm: RFdtreefit(X, y, loss, max_features, min_samples_leaf)
2     if |X| < min_samples_leaf then return Leaf(y)
3     col, split = RFbestsplit(X, y, loss, max_features)
4     if col = -1 then return Leaf(y)
5     lchild = fit(X[X ≤ split], y[X ≤ split])
6     rchild = fit(X[X > split], y[X > split])
7     return DecisionNode(col, split, lchild, rchild)
```

Same as decision tree except we pass max_features to Rfbestsplit()

Finding best split in decision node in RF

1 **Algorithm:** *RFbestsplit(X, y, loss, max_features)*
2 $best = (col = -1, split = -1, loss = loss(y))$
3 $vars =$ pick $max\_features$ variables from all p
4 **for** $j \in vars$ **do**
5 $candidates =$ pick 11 values from $X_{-,j}$
6 **foreach** $split \in candidates$ **do**
7 $yl = y[X \leq split]$
8 $yr = y[X > split]$
9 **if** $|yl| = 0$ *or* $|yr| = 0$ **then continue**
10 $l = \frac{loss(yl)+loss(yr)}{2}$
11 **if** $l = 0$ **then return** $col, split$
12 **if** $l < best[loss]$ **then** $best = (col, split, l)$
13 **end**
14 **end**
15 **return** *col, split*

Only diff with decision tree

Pick, say, 11 not all possible X values. We get better generality and code is much faster!

Should pick midpoint between split value and next smallest X

Simplest RF prediction (ESLII p588)

- But doesn't use all information to make best prediction
- Need to use weighted averages / votes

Regression: $\hat{f}_{\text{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^{B}(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

RF prediction

Weighted average of y values among the leaves reached by running x down each tree

Algorithm: $RFpredict_{regr}(\{T_1..T_{ntrees}\},\ x)$

- **Let** $leaf_i = leaf(T_i.root, x)$
- $nobs = \sum_{i=1}^{ntrees} |leaf_i.y|$
- $ysum = \sum_{i=1}^{ntrees} \sum_j leaf_i.y_j$
- **return** $ysum/nobs$

Weight each leaf's vote by |y| among the leaves reached by running x down each tree

Algorithm: $RFpredict_{class}(\{T_1..T_{ntrees}\},\ x)$

- **foreach** $i = 1..ntrees$ **do**
  - $leaf = leaf(T_i.root, x)$
  - $y' = \text{mode}(leaf.y)$
  - $counts[y'] \mathrel{+}= |leaf.y|$
- **end**
- **return** $\text{argmax}(counts)$