Forests and Other Ensembles

36-462/662, Spring 2022

1 March 2022 (Lecture 13)

Contents

Trees

Recap on the CART procedure for growing trees

- If there are less than n_{min} cases left, stop
- Consider all possible binary splits based on *one* feature alone
  - For regression, consider the reduction in $\mathrm{Var}[Y|X]$
  - For classification outcomes, consider the reduction in $H[Y|X]$
- Does the best split improve the prediction by at least δ?
  - If no, stop
  - If yes, recursively apply the procedure to cases on either side of the split
- Prune the tree by cross-validation

Virtues and limits of trees

- Trees are fast to fit, fast to make predictions from, and east for people to grasp
  - At least, all those things are true of *small* trees, with few leaves
- Trees can approximate any function you like, as closely as you like
  - Proof: It's what you do to plot a curve on a screen with pixels
- Fully nonparametric, perfectly happy to handle interactions
  - In fact, by default every variable gets to "interact" with every other
- Close approximation may nonetheless need a really large number of leaves
- More leaves means less bias but also more variance
  - True even if the splits are all treated as fixed
  - Even more true if the splits are found by adaptively growing the tree

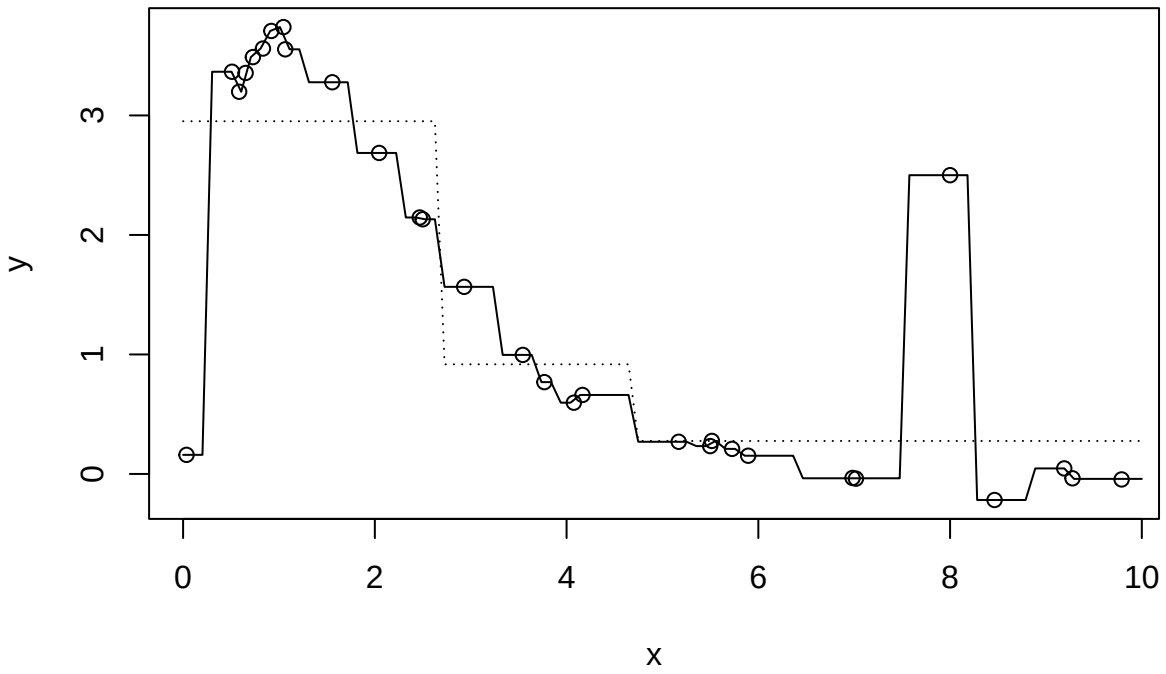

Figure 1: Some simulated regression data (dots), plus a regression tree fit to the data by growing a big tree without pruning (solid line), and another regression tree fit with the default control settings (dotted line). Notice how the more stable, dotted-line tree misses the outlier (if it_ is *an outlier and not a genuine feature of the data-generating process), but also misses some of the apparent structure of the regression curve (if it* is _structure and not just seeing patterns in noise).

- Getting really good performance may need a really big tree (with low bias), but those have very high variance
- How can we grow big trees with low variance?

Forests

- "Forest" methods combine the predictions from multiple trees
- Forests are a special case of **ensemble methods**, which combine multiple models to improve on what any one of them could do
- The simplest sort of combination is averaging
  - For classifers, "averaging" = voting
  - (Or, if you like, average the conditional probabilities and then threshold the average probability)

Three leading forms of ensemble methods

1. **Bagging**: randomly perturb the data, grow a tree to the new data, average
2. **Random forests**: combine bagging with *random* feature selection
3. **Boosting**: sequentially fit models to the *errors* of earlier models

Bagging, or "bootstrap averaging" (Breiman 1996)

The bagging procedure is simplicity itself:

- Start with a data set $D = (X_1, Y_1), \ldots (X_n, Y_n)$
- Fix the number of trees m we want in the forest
- For $k \in 1 : m$
  - Generate $\tilde{D}_k$ by resampling the (X_i, Y_i) n times, with replacement
    - That is, $\tilde{D}_k$ is a resampling ("nonparametric") bootstrap simulation of the data-generating process
    - With high probability, some data points are repeated in $\tilde{D}_k$, and some do not appear at all
  - Grow a tree τ_k from $\tilde{D}_k$
    - Typically without pruning
- Make predictions by averaging the τ_k

A little demo

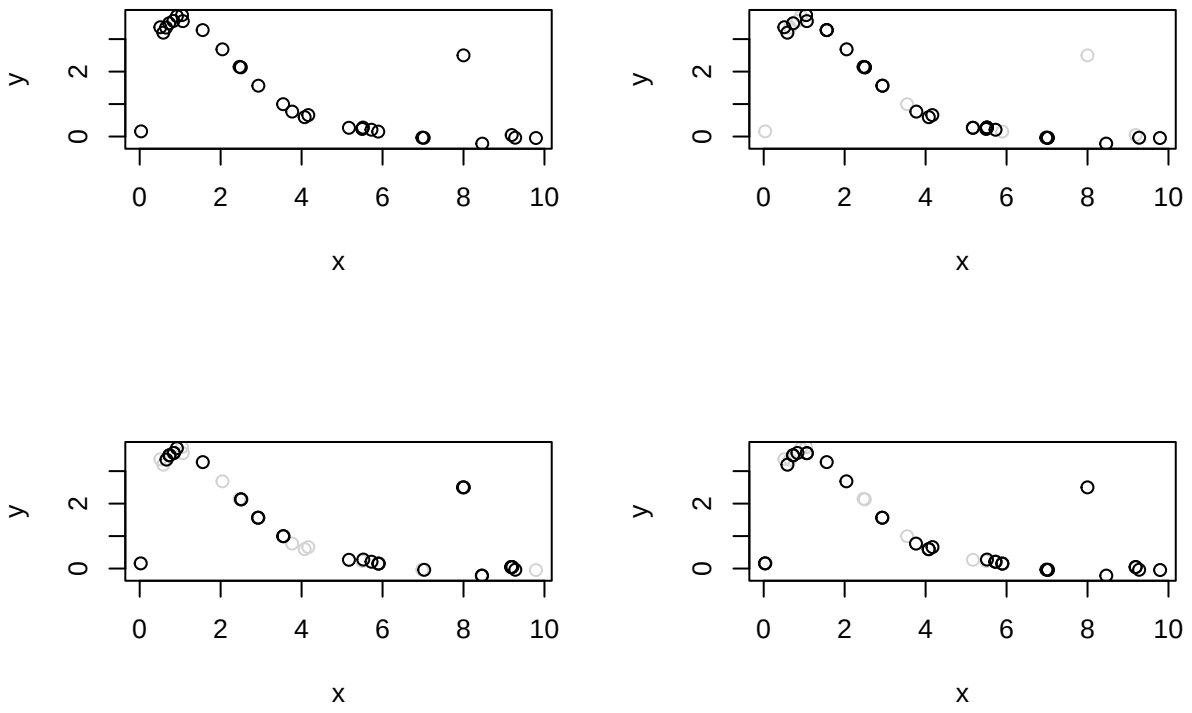

Figure 2: Original data for the running example (top left) and three bootstrap resamplings; in each resampling, the full data set is shown in light grey (for comparisons), and the coordinates are slightly "jittered", so that a repeatedly-sampled point appears as multiple points very close to each other.

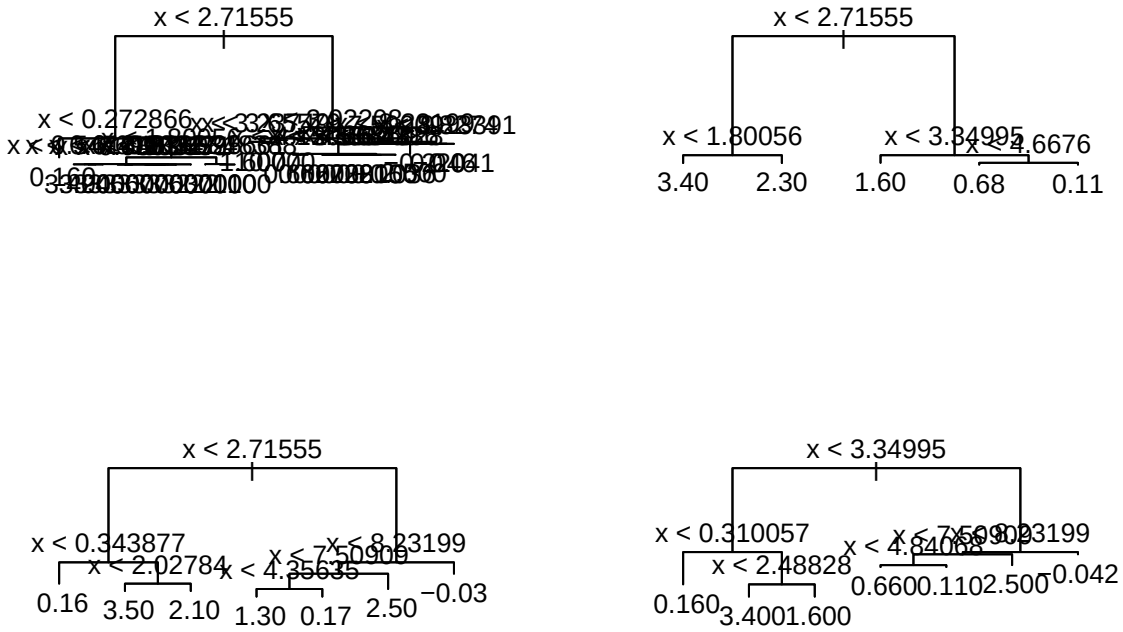

Figure 3: Tree fit to the full data (top left), plus the three trees fit to the three bootstrap resamplings from the previous figure.

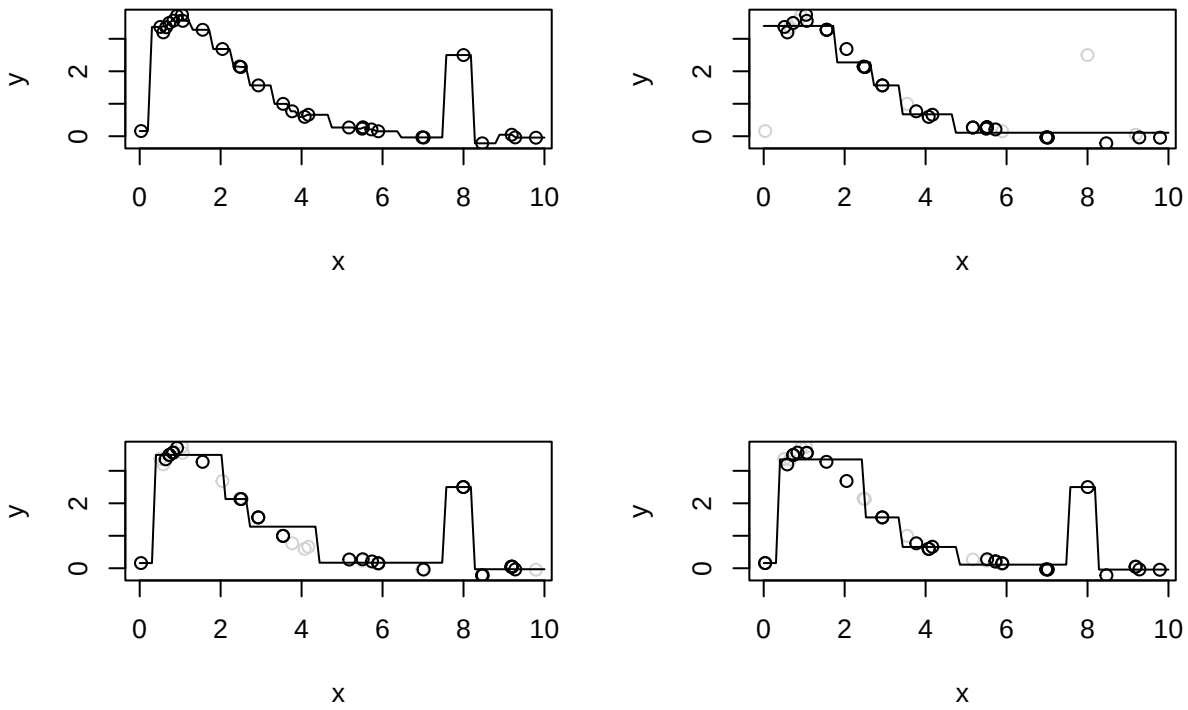

Figure 4: Original data (top left), plus the same three resamplings, with the regression function estimated by the tree fit to each data set.

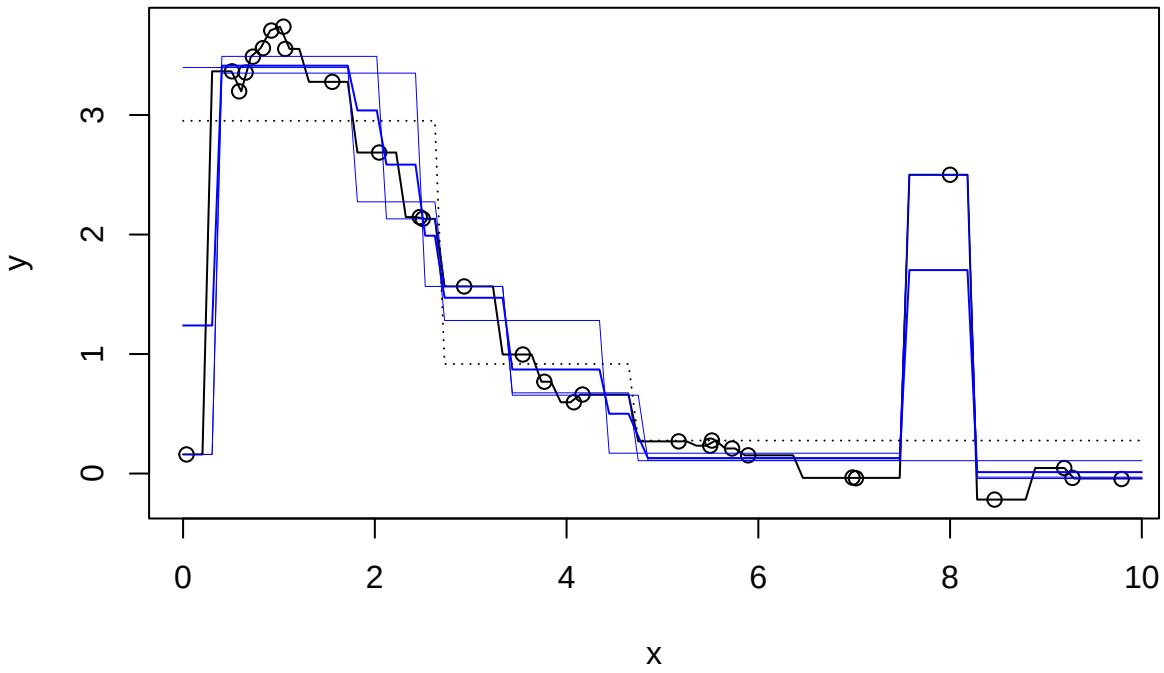

Figure 5: Full data (points), plus regression tree fit to them (black), plus the three trees fit to bootstrap resamplings (thin blue lines), plus the average of the three bootstrapped trees, i.e., the bagged model (thick blue line). Notice how the impact of the outlier is attenuated by bagging, but the main features of the unstable-but-sensitive big tree have been preserved, and the bagged curve show more detail than just fitting a stable-but-insensitive tree (dotted black line) — for instance, bagging picks up that the curve rises for small values of x.

Why does this help?

- Bootstrapping perturbs the data, but in ways which resemble re-running the experiment
- Even if each tree τ_k is over-fit to $\tilde{D}_k$, what they have in common will be the main features of the data set; what's peculiar to each resampling will tend to average out
- So the trees shouldn't be (much) more biased than a tree fit to the whole data, but with less variance, or sensitivity to accidents of the original data

Why it might be a bit surprising that bagging helps

Making predictions using a bag of trees is equivalent to making predictions using one giant tree with a huge number of leaves. (Can you prove this, and explain why if each tree has r leaves, a forest of m trees is equivalent to one tree with $O(r^m)$ leaves?) And we know that giant trees should be really unstable, with really high variance. So it might seem that we haven't gained anything, but we really do get better, more stable predictions from bagging. The trick is that we don't get our predictions by growing just *any* giant tree — only trees that arise by averaging many smaller trees are allowable. This however suggests that we should be really careful about what we mean when we say things like "simpler models are usually better", or even "simpler models are usually more stable" (Domingos 1999b, 1999a).

How big a forest?

- m in the range of 100–1000 is often plenty
- Statistically, increasing m can only reduce the variance
- But there are diminishing returns to increasing m, for an interesting reason...

Some basic math about averaging

- Suppose we've got two random variables, let's say T_1 and T_2, with equal variance $\sigma^2 > 0$, and correlation $\rho > 0$
- What's $\mathrm{Var}\left[(T_1 + T_2)/2\right]$?

$$\mathrm{Var}\left[\frac{T_1 + T_2}{2}\right] = \frac{1}{4}\mathrm{Var}\left[T_1 + T_2\right] \quad (1)$$

$$= \frac{1}{4}\left(\mathrm{Var}\left[T_1\right] + \mathrm{Var}\left[T_2\right] + 2\mathrm{Cov}\left[T_1, T_2\right]\right) \quad (2)$$

$$= \frac{\sigma^2}{2} + \frac{2\rho\sigma^2}{4} = \frac{\sigma^2}{2}(1+\rho) \quad (3)$$

  - If $\rho = 0$, we get the familiar result, that averaging two variables halves the variance
  - If $\rho > 0$, we get more variance than that, by a factor of ρ
    - If T_1 is above [below] its mean, T_2 tends to also be above [below] its mean, and so fluctuations for the average are exaggerated
  - If $\rho < 0$, we'd get *less* variance than if they were uncorrelated
    - If one variable is above its mean the other tends to be below its mean, suppressing fluctuations for the average
- If we average m variables $T_1, \ldots T_m$, all with variance σ^2 and all with correlation ρ with each other, we

get

$$\begin{aligned}
\mathrm{Var}\left[m^{-1}\sum_{k=1}^{m} T_k\right] &= \frac{\sigma^2}{m} + \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l\neq k}\mathrm{Cov}\left[T_k, T_l\right] &(4)\\
&= \frac{\sigma^2}{m} + \frac{m(m-1)}{m^2}\rho\sigma^2 &(5)\\
&= \sigma^2\left(\frac{1}{m} + \rho\frac{m-1}{m}\right) &(6)
\end{aligned}$$

- Even as $m \to \infty$, this goes to $\rho\sigma^2$, not to 0
  - Unless $\rho = 0$
  - ρ can't be negative if $m \to \infty$ (see backup)

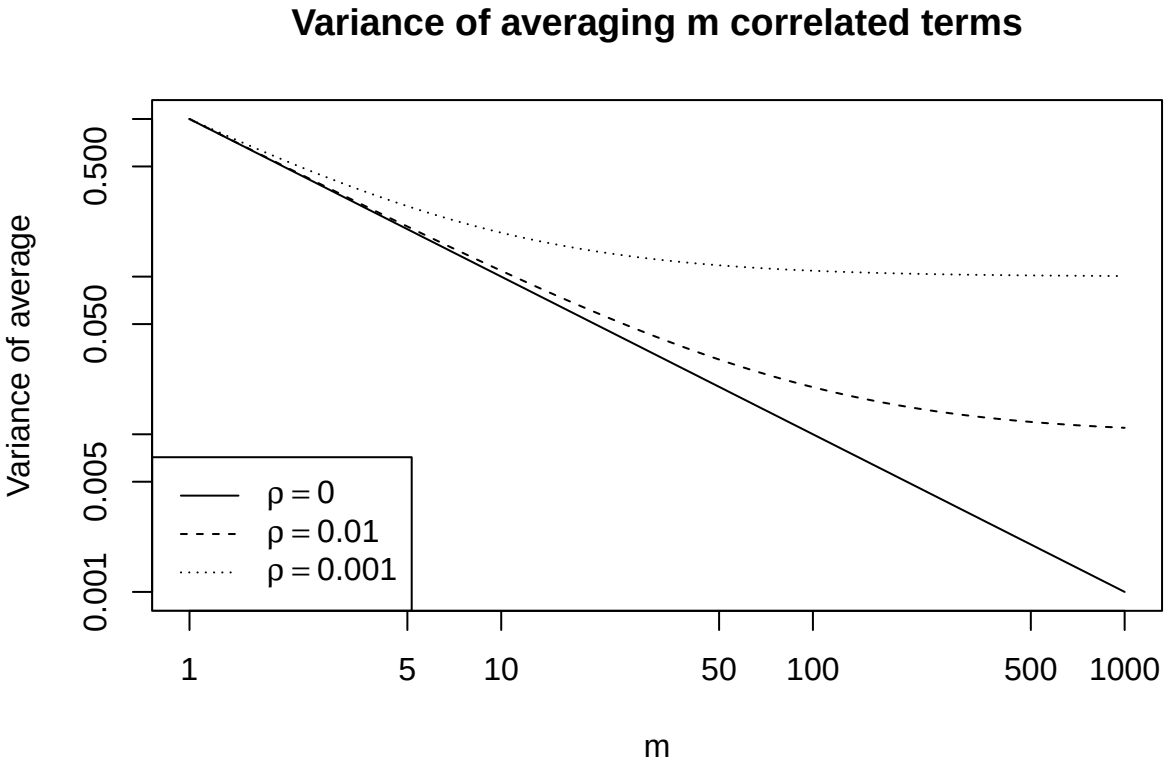

Figure 6: Variance of averaging m terms of equal variance $\sigma^2 = 1$, each with correlation ho with the others. Notice that the variance declines monotonically with m, but, unless $ho = 0$, it asymptotes to a non-zero value.

Why this matters here

- Each tree from bagging has the same distribution
  - τ_k is a random tree, because $\tilde{D}_k$ is a random data set
  - All bootstraps have the same distribution, so τ_k and τ_l have the same distribution
  - For any point x, $T_k \equiv \tau_k(x)$ is random variable
  - T_k will have some variance σ^2
- Trees will be positively correlated
  - i.e., if $\tau_k(x) > \mathbb{E}\left[\tau(x)\right]$, then we should tend to expect $\tau_l > \mathbb{E}\left[\tau(x)\right]$
    - Whatever quirk of the initial sample D led to over-estimating the function at x on run k, it's probably shared with other resamplings of D
    - (Conditional on D, the τ_k are IID)

- All trees are equally correlated
  - In the sense that $\mathrm{Cov}\left[\tau_k(x), \tau_l(x)\right]$ has to be the same for all $k \neq l$
    - Could change with x
  - This follows because, given D, all the resamplings $\tilde{D}_k$ and $\tilde{D}_l$ are IID
- So at any point x, $\tau_k(x)$ has some variance $\sigma^2(x)$ and some correlation $\rho(x)$ across trees, and so

$$\mathrm{Var}\left[\frac{1}{m}\sum_{k=1}^{m}\tau_k(x)\right] = \sigma^2(x)\left(\frac{1}{m} + \rho(x)\frac{m-1}{m}\right) \to \sigma^2(x)\rho(x)$$

  - (Resampling the same fixed data set over and over can't give us unlimited information)

Random forests

What can we do to reduce the correlation between trees?

- Tweak the data
  - But bootstrap is so natural...
- Add noise to the prediction
  - Seems drastic...
- "Random forests" (Breiman 2001)
  - Paper considered a whole range of ways of randomly building lots of trees, including bagging
  - Introduced the idea of combining bagging with **random feature selection**
  - We now call that combination "random forests"

The random forests algorithm of Breiman (2001)

- Given:
  - Data set $D = (X_1, Y_1), \ldots (X_n, Y_n)$
  - Desired forest size m
  - Number of features to consider for each split $q < p = dim(X)$
- repeat m times:
  - Generate a bootstrap sample $\tilde{D}_k$
  - Run CART to build a tree, but at each interior node, pick q features from X, and *only* consider splits on those features
    - Re-select features independently at each node, with no memory of what was done in other trees, or elsewhere in this tree
    - Don't bother pruning
  - Return tree τ_k
- To predict, average over the τ_k

Some practicalities

- Number of features to pick q shouldn't be too big
  - $q = \sqrt{p}$ is a common default and often works pretty well
  - I am surprised at how often $q = 1$ is competitive
  - Of course, if $p = 1$ to start with, random forests becomes good old-fashioned bagging
    - So there's nothing for me to show you here with the running example
- Number of trees: $m = 100$ or $m = 1000$ often plenty
  - As with bagging, increasing m can only improve things statistically, *but* there are diminishing returns, and computational costs
- Common: re-estimate the predictions in each leaf by take the data points *not* included in the tree's bootstrap sample (the "out-of-bag" data points) and "dropping them down the tree"
  - Reduces over-fitting a bit

What random forests does

- Different trees will make different random feature selections
- This keeps them from being *too* similar
- Using bootstrapped versions of the same data forces them to all be trying to solve the same prediction problem
- Results in a good balance of not-too-much bias, not-too-much variance, and low(er) correlation across trees
- RF is simple, fast, and often very competitive in terms of sheer predictive performance

– Computation: CART is fast, bootstrapping is fast, growing the forest is “embarrassingly parallel”
– Sheer performance is often close to the best available with much more computational effort

Boosting

- Bagging and random forests are parallel: no communication or cross-talk between growing tree 1 and tree 2, etc.
- **Boosting** is a **sequential** procedure, where tree k is grown in a way that tries to compensate for the mistakes of trees $1, 2, \ldots k-1$

The basic boosting procedure

- Given: data set $D = (X_1, Y_1), \ldots (X_n, Y_n)$, number of rounds of boosting m
  - You can make m adapt to the data if you really want
- Initialize all data points to equal weight $w_i^{(1)} = 1$ for $i \in 1:n$
- for $k \in 1:m$
  - Grow tree τ_k by doing a *weighted* fit to the data, with weights $w^{(k)}$
  - Calculate the error τ_k makes on each data point, e_{ik}
  - Multiply weights, so $w_i^{(k+1)} = w_i^{(k)} f(e_{ik})$ for some function f
    - f should increase with the size of the error, and be close to or actually 0 for 0 error
    - Good choices of f depend on whether we're doing classification or regression, etc.
- Return the set of trees τ_k
- Make predictions by averaging over the trees
  - Possibly weight tree τ_k by how well it fit the data

What boosting does

- The first tree tries to fit all the data equally
- The *second* tree doesn't care much about data points that the first tree fit. Instead, it adjusts itself to try to fit the errors of tree 1. (Weights go up with error.)
- The *third* tree doesn't care much about data points that *either* of the first two trees fit, and *really* doesn't care about data points that *both* of them fit, but focuses on data points *neither* of them fit
- And so on for the fourth, seventh, 183rd tree...
- Each model in the sequence tries to correct the errors of the models that came before it

Relationship between trees in boosting

- Using the same data...
- But with very different weights...
- Weight moves to places the earlier trees didn't predict well, which can create a kind of anti-correlation in the predictions
- Not easily analyzed with variances and correlations (unlike bagging)
  - Trees are dependent, changing variance, changing correlations
  - Weight tends to accumulate on points which are very hard for *any* tree to predict
  - Think of boosting as a game between a Learner and an Adversary
    - Learner scores points if it can grow a tree that fits all the data
    - Adversary scores points if it can propose points that give Learner trouble
    - Game converges to an equilibrium or stale-mate

In practice...

- Boosting typically uses **weak learners**
  - Say, a tree with only two leaves
  - Highly biased, but very stable

- Combining many *different* weak learners creates a strong learner
  - Boosting specifically uses each learner to compensate for the faults of the others
- Use a lot of rounds of boosting
  - Each weak learner is easy to fit, after all...
- Often highly competitive performance
  - Maybe not *quite* as good as random forests

Note on regression boosting and gradient boosting

- For regression, boosting is (often) equivalent to fitting a model, and then fitting a model to the residuals of the first model, and then fitting a model to the residuals of the second model, etc., and adding them together (Hastie, Tibshirani, and Friedman 2009)

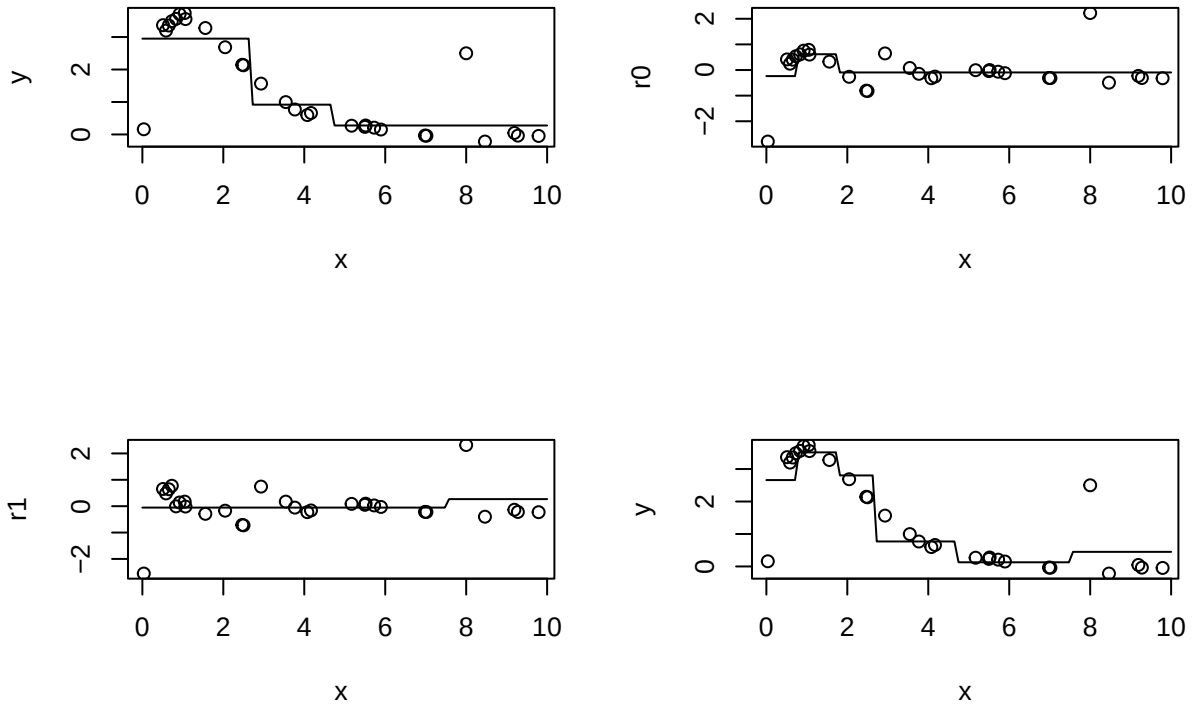

Figure 7: Illustration of boosting. Top left: running-example data, plus a regression tree with only three leaves. Top right: Residuals from the first model, plus a three-leaf tree fit to those residuals. Bottom left: residuals from the second model, plus a three-leaf tree fit to *those* residuals. Bottom right: Original data again, plus the sum of the three estimated trees. In practice, one would use many more than three steps of boosting.

- This idea can be generalized to other loss functions than squared error, with the derivative of the loss taking the role of the residuals; this is **gradient boosting**
  - Remember from HW 5 Q2d that when we worked out the gradient of the MSE for linear regression, we got a weighted sum of the residuals...

Not just trees

- Bagging and boosting are often applied to all kinds of models, not just CART
  - Random forests are harder to combine with other models, because what's the equivalent of splitting on features?
- There's no point to using bagging with linear estimators
  - E.g., ordinary least squares for a linear regression
  - Because: taking $m \to \infty$ just gives you back the linear estimator on the full data!
- Bagging tends to improve very nonlinear but unstable methods
  - Like trees
  - Or like linear regression *with variable selection*
- Boosting tends to improve nonlinear, low-capacity methods
  - Like trees with very few leaves
  - Or simple smoothing methods

Diversity in ensemble methods (after Krogh and Vedelsby (1995))

- Think about regression for the moment
- We're trying to estimate μ, and we have (for simplicity) *two* estimates, T_1 and T_2
  - Our average estimate is $\overline{T} \equiv \frac{T_1+T_2}{2}$
  - The variance around this average estimate is V

$$
\begin{aligned}
V &\equiv \frac{1}{2}\left[(T_1-\overline{T})^2 + (T_2-\overline{T})^2\right] && (7)\\
&= \frac{1}{2}\left[(T_1-\mu+\mu-\overline{T})^2 + (T_2-\mu+\mu-\overline{T})^2\right] && (8)\\
&= \frac{1}{2}\left[(T_1-\mu)^2 + (T_2-\mu)^2 + 2(\mu-\overline{T})^2 + (2(T_1-\mu)+2(T_2-\mu))(\mu-\overline{T})\right] && (9)\\
&= \frac{1}{2}\left[(T_1-\mu)^2 + (T_2-\mu)^2 + 2(\mu-\overline{T})^2 + 2(T_1+T_2-2\mu)(\mu-\overline{T})\right] && (10)\\
&= \frac{1}{2}\left[(T_1-\mu)^2 + (T_2-\mu)^2 + 2(\mu-\overline{T})^2 + 4(\overline{T}-\mu)(\mu-\overline{T})\right] && (11)\\
&= \frac{1}{2}\left[(T_1-\mu)^2 + (T_2-\mu)^2 - 2(\mu-\overline{T})^2\right] && (12)\\
&= \frac{(T_1-\mu)^2 + (T_2-\mu)^2}{2} - (\mu-\overline{T})^2 && (13)\\
(\mu-\overline{T})^2 &= \frac{(T_1-\mu)^2 + (T_2-\mu)^2}{2} - V && (14)
\end{aligned}
$$

- This generalizes to more than two estimates, as you'll see in the homework.
- The final left-hand side the squared error of the average estimator
  - The RHS is the averaged squared error of the different estimators...
  - ... *minus* the variance of the estimators
  - All else being equal, better individual models will improve the ensemble
  - All else being equal, *more diverse* individual models will improve the ensemble
- In a slogan,

$$\text{(performance of group)} = \text{(average individual performance)} + \text{(diversity of group)}$$

Some notes on this math

- This works equally well if you're doing a weighted average of predictors — you just need to define V as the weighted variance within the ensemble (exercise!)

- Similar math for any loss function that has a bias-variance decomposition
  - The variance V here is the variance across the ensemble, not the variance which was going to $\rho\sigma^2$ as m grew above

Further reading

- Look at the suggested readings from Berk (2008) and Hastie, Tibshirani, and Friedman (2009) for technical details on bagging, random forests and boosting
- Schapire and Freund (2012) is a great reference on boosting, and one of the best-written books on machine learning I've ever read
- On diversity, see Page (2007) (especially on how efforts to get the "best" individuals can reduce the group diversity and therefore lower performance)

Backup topics

The minimum (average) correlation among m variables

Suppose we have m variables $T_1, \ldots T_m$, each with variance σ^2, and $\mathrm{Cov}\left[T_k, T_l\right] = \rho_{kl}\sigma^2$. The variance of the average is

$$\begin{aligned}
\mathrm{Var}\left[\frac{1}{m}\sum_{k=1}^{m} T_k\right] &= \frac{\sum_{k=1}^{m}\mathrm{Var}\left[T_k\right]}{m^2} + \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l\neq k}\mathrm{Cov}\left[T_k, T_l\right] && (15)\\
&= \frac{\sigma^2}{m} + \frac{\sigma^2}{m^2}\sum_{k=1}^{m}\sum_{l\neq k}\rho_{kl} && (16)\\
&= \sigma^2\left(\frac{1}{m} + \frac{m(m-1)}{m^2}\overline{\rho}\right) && (17)
\end{aligned}$$

where $\overline{\rho}$ is the average of all the ρ_{kl}. Since this is a variance, it must be ≥ 0, which implies

$$\begin{aligned}
\frac{1}{m} + \frac{m-1}{m}\overline{\rho} &\geq 0 && (18)\\
(m-1)\overline{\rho} &\geq -1 && (19)\\
\overline{\rho} &\geq \frac{-1}{m-1} && (20)
\end{aligned}$$

Notes/exercises:

0. Since this is the minimum average correlation, if all variables have equal correlation ρ, this is also the minimum value of ρ.
1. The assumption that all T_k have the same variance σ^2 is not essential — it just simplifies the book-keeping. (Going through the book-keeping will be a character-building exercise.)
2. You should be able to create a family of m variables with equal variance, and with correlation exactly $-1/(m-1)$, *without* Googling...

Exchangeable variables and random limits for averages

- A sequence of random variables $T_1, T_2, \ldots T_m, \ldots$ is **exchangeable** when the distribution doesn't change if we permute them
  - All IID sequences are exchangeable, but not vice versa
  - Every *infinite* exchangeable sequence is a mixture of IID sequences
    - Meaning: $T_k \perp T_l | D$ for some random variable D
    - For us, bootstrap resamplings $\tilde{D}_k$ are exchangeable, and IID condtional on the *original* sample D
- Exchangeability implies that (T_k, T_l) has the same distribution as (T_1, T_2) for any $i \neq j$
  - (Can you prove this?)
- This in turn implies that $\mathrm{Cov}\left[T_k, T_l\right] = \rho\sigma^2$, where $\sigma^2 = \mathrm{Var}\left[T_k\right]$ and ρ is some constant correlation
- So, as we've seen,
$$\mathrm{Var}\left[\frac{1}{m}\sum_{k=1}^{m} T_k\right] \to \rho\sigma^2$$
- But $\mathbb{E}\left[m^{-1}\sum_k T_k\right] = \mathbb{E}\left[T_1\right]$
- $m^{-1}\sum_k T_k$ converges, but it converges to a *random* limit, and the variance of those limits is $\rho\sigma^2$
- For much, much more, see Kallenberg (2005)

The name "boosting"

- The name "boosting" comes from the idea of "boosting a weak learner to a strong learner"
- The original idea was (see, e.g., Kearns and Vazirani (1994)):
  - We're trying to classify balanced data (with $Y = 0$ and $Y = 1$ equally probable), and perfect classification is possible, if only we knew the right rule
  - We can be (say) 90% confident that our "weak learner", trained on n_0 data points, will classify with accuracy (say) 51%, and won't get better with more data
    - (Exact numbers don't matter, just: only a little better than chance, with only a fixed confidence level)
  - Now collect $n = mn_0$ data points, break it up randomly into m chunks, and run the weak learner m times, and use the majority vote
  - By taking m big enough, we be arbitrarily confident that the majority vote has an arbitrarily low error rate
    - More formally, for any $\epsilon > 0$, $\delta > 0$, there is an m such that with at least mn_0 data points, the majority vote has an error rate $\leq \epsilon$ with probability at least $1 - \delta$
- This is actually closer to bagging than to modern boosting...

References

Berk, Richard A. 2008. *Statistical Learning from a Regression Perspective*. New York: Springer-Verlag.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24:123–40.

———. 2001. "Random Forests." *Machine Learning* 45:5–32. https://doi.org/10.1023/A:1010933404324.

Domingos, Pedro. 1999a. "Process-Oriented Estimation of Generalization Error." In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 714–19. San Francisco: Morgan Kaufmann. http://www.cs.washington.edu/homes/pedrod/papers/ijcai99.pdf.

———. 1999b. "The Role of Occam's Razor in Knowledge Discovery." *Data Mining and Knowledge Discovery* 3:409–25. http://www.cs.washington.edu/homes/pedrod/papers/dmkd99.pdf.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second. Berlin: Springer. http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

Kallenberg, Olav. 2005. *Probabilistic Symmetries and Invariance Principles*. New York: Springer-Verlag.

Kearns, Michael J., and Umesh V. Vazirani. 1994. *An Introduction to Computational Learning Theory*. Cambridge, Massachusetts: MIT Press.

Krogh, Anders, and Jesper Vedelsby. 1995. "Neural Network Ensembles, Cross Validation, and Active Learning." In *Advances in Neural Information Processing Systems 7 [Nips 1994]*, edited by Gerald Tesauro, David Tourtetsky, and Todd Leen, 231–38. Cambridge, Massachusetts: MIT Press. https://papers.nips.cc/paper/1001-neural-network-ensembles-cross-validation-and-active-learning.

Page, Scott E. 2007. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton, New Jersey: Princeton University Press. https://doi.org/10.2307/j.ctt7sp9c.

Schapire, Robert E., and Yoav Freund. 2012. *Boosting: Foundations and Algorithms*. Cambridge, Massachusetss: MIT Press.