Bagging

Ryan Tibshirani
Data Mining: 36-462/36-662

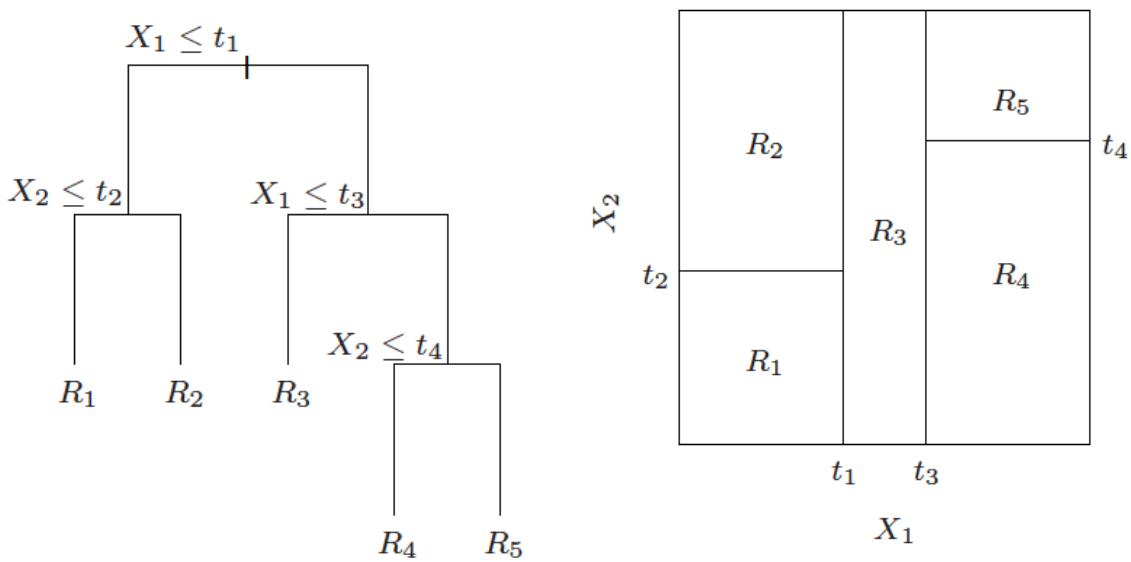
April 23 2013

Optional reading: ISL 8.2, ESL 8.7

Reminder: classification trees

Our task is to predict the class label $y \in \{1, \ldots K\}$ given a feature vector $x \in \mathbb{R}^p$. Classification trees divide the feature space $\mathbb{R}^p$ up into several rectangles, and then assign to each rectangle R_j a particular predicted class c_j:

$$\hat{f}^{\text{tree}}(x) \;=\; \sum_{j=1}^{m} c_j \cdot 1\{x \in R_j\} \;=\; c_j \text{ such that } x \in R_j$$



Given training data (x_i, y_i), $i = 1, \ldots n$, with $y_i \in \{1, \ldots K\}$ being the class label and $x_i \in \mathbb{R}^p$ the associated feature vector, the CART algorithm successively splits the features in a greedy fashion

Its strategy is to grow a large tree and then prune back using cross-validation. At the end, in each rectangle R_j the predicted class is simply the majority class:

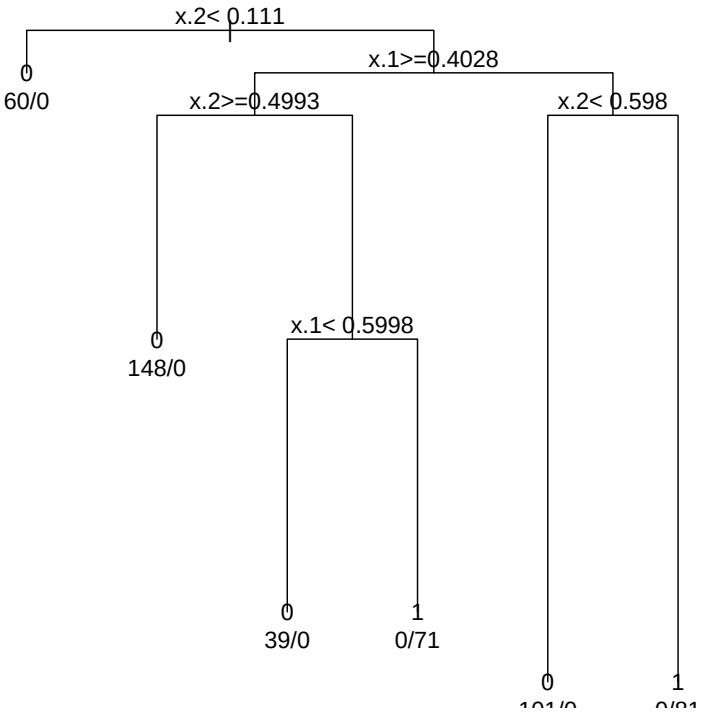
$$c_j = \underset{k=1,\ldots K}{\operatorname{argmax}} \ \hat{p}_k(R_j)$$

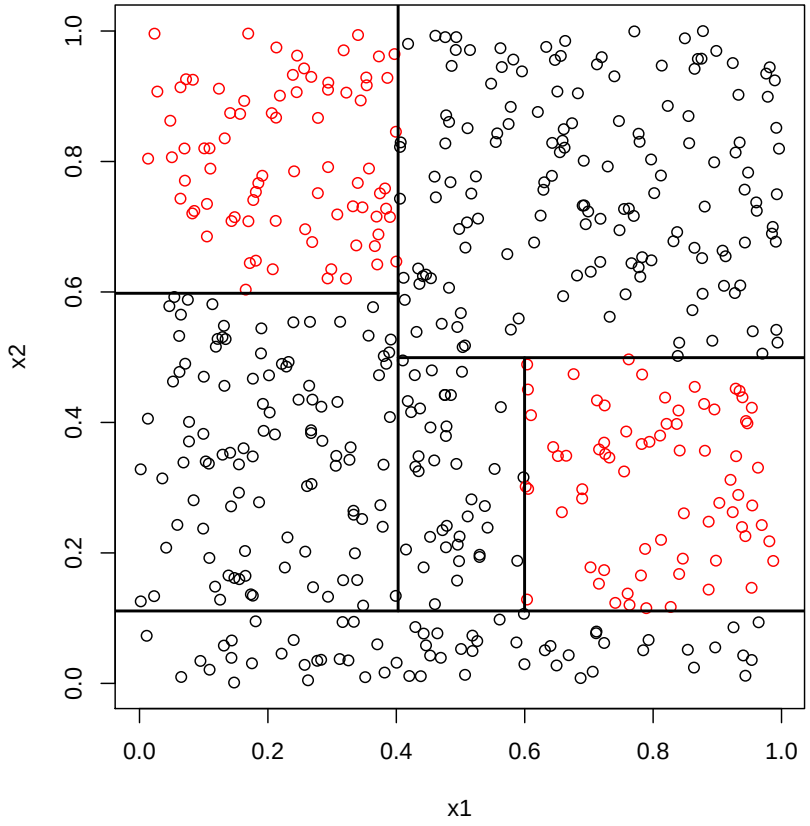
where $\hat{p}_k(R_j)$ is the proportion of points of class k that fall into region R_j:

$$\hat{p}_k(R_j) = \frac{1}{n_j} \sum_{x_i \in R_j} 1\{y_i = k\}$$

This gives us predicted class probabilities for each region

Example: $n = 500$ points in $p = 2$ dimensions, falling into classes 0 and 1 (as marked by colors)





Classification trees are popular since they are highly interpretable. They are also model-free (don't assume an underlying distribution for the data)

But they don't generally give a prediction accuracy competitive with that of other classifiers (logistic regression, k-nearest-neighbors, etc.) The reason: trees have somewhat inherently high variance. The separations made by splits are enforced at all lower levels of the tree, which means that if the data is perturbed slightly, the new tree can have a considerably different sequence of splits, leading to a different classification rule

In this lecture (and the next) we'll learn of two ways to control the variance, or stabilize the predictions made by trees. Of course these solutions aren't perfect: in doing so, we can greatly improve prediction accuracy but we suffer in terms of interpretability

Joint distribution and Bayes classifier

Suppose that we observe training data (x_i, y_i), $i = 1, \ldots n$, which represents n independent draws from some unknown probability distribution $\mathcal{F}$. E.g., this could be classification data, with $y_i \in \{1, \ldots K\}$ being the class label and $x_i \in \mathbb{R}^p$ the associated feature vector

Note that $\mathcal{F}$ describes the joint distribution of X and Y:

$$\mathrm{P}_{\mathcal{F}}\big\{(X, Y) = (x, y)\big\}$$

Recall that if we knew $\mathcal{F}$, then the best thing to do would be to simply classify according to the Bayes classifier:

$$\begin{aligned} f(x) &= \underset{j=1,\ldots K}{\operatorname{argmax}} \ \mathrm{P}_{\mathcal{F}}(Y = j | X = x) \\ &= \underset{j=1,\ldots K}{\operatorname{argmax}} \ \mathrm{P}_{\mathcal{F}}\big\{(X, Y) = (x, j)\big\} \end{aligned}$$

The bootstrap

The bootstrap[1] is a fundamental resampling tool in statistics. The basic idea underlying the boostrap is that we can estimate the true $\mathcal{F}$ by the so-called empirical distribution $\hat{\mathcal{F}}$

Given the training data (x_i, y_i), $i = 1, \ldots n$, the empirical distribution function $\hat{\mathcal{F}}$ is simply

$$P_{\hat{\mathcal{F}}}\{(X, Y) = (x, y)\} = \begin{cases} \frac{1}{n} & \text{if } (x, y) = (x_i, y_i) \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

This is just a discrete probability distribution, putting equal weight $(1/n)$ on each of the observed training points

[1] Efron (1979), "Bootstrap Methods: Another Look at the Jacknife"

A bootstrap sample of size m from the training data is

$$(x_i^*, y_i^*),\ i = 1, \ldots m$$

where each (x_i^*, y_i^*) are drawn from uniformly at random from $(x_1, y_1), \ldots (x_n, y_n)$, with replacement

This corresponds exactly to m independent draws from $\hat{\mathcal{F}}$. Hence it approximates what we would see if we could sample more data from the true $\mathcal{F}$. We often consider $m = n$, which is like sampling an entirely new training set

Note: not all of the training points are represented in a bootstrap sample, and some are represented more than once. When $m = n$, about 36.8% of points are left out, for large n (Homework 6)

Bagging

Given a training data (x_i, y_i), $i = 1, \ldots n$, bagging[2] averages the predictions from classification trees over a collection of boostrap samples. For $b = 1, \ldots B$ (e.g., $B = 100$), we draw n boostrap samples (x_i^{*b}, y_i^{*b}), $i = 1, \ldots n$, and we fit a classification tree $\hat{f}^{\text{tree},b}$ on this sampled data set. Then at the end, to classify an input $x \in \mathbb{R}^p$, we simply take the most commonly predicted class:

$$\hat{f}^{\text{bag}}(x) = \underset{k=1,\ldots K}{\text{argmax}} \sum_{b=1}^{B} 1\{\hat{f}^{\text{tree},b}(x) = k\}$$

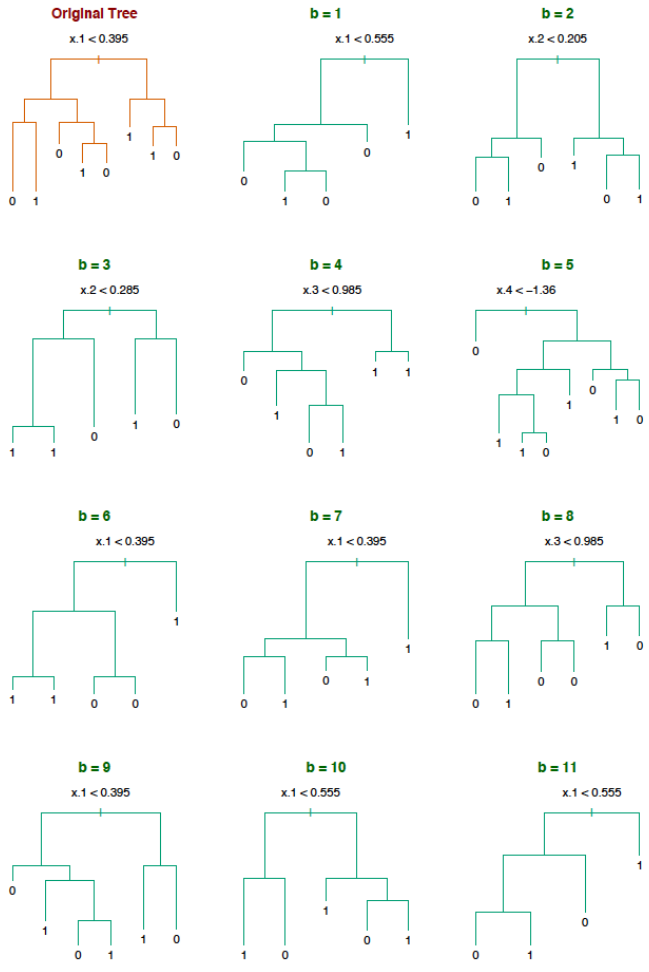
This is just choosing the class with the most votes. Two options:

- Simple strategy: grow fairly large trees on each sampled data set, with no pruning
- More involved strategy: prune back each tree as we do with CART, but use the original training data (x_i, y_i), $i = 1, \ldots n$ as the validation set, instead of performing cross-validation

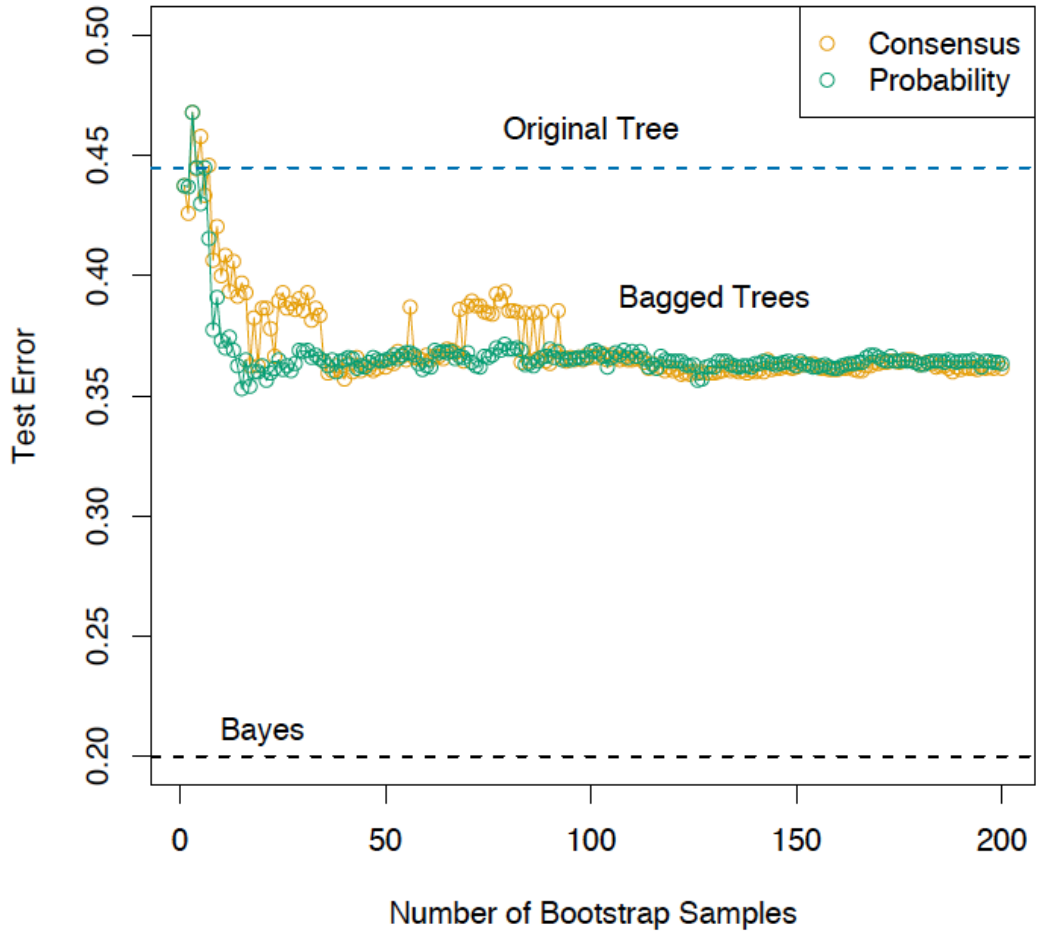
[2] Breiman (1996), "Bagging Predictors"

Example: bagging

Example (from ESL 8.7.1): $n = 30$ training data points, $p = 5$ features, and $K = 2$ classes. No pruning used in growing trees:



Bagging helps decrease the misclassification rate of the classifier (evaluated on a large independent test set). Look at the orange curve:



Example: Breiman's bagging

Example from the original Breiman paper on bagging: comparing the misclassification error of the CART tree (pruning performed by cross-validation) and of the bagging classifier (with $B = 50$):

Data Set	$\bar{e}_S$	$\bar{e}_B$	**Decrease**
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Voting probabilities are not estimated class probabilities

Suppose that we wanted estimated class probabilities out of our bagging procedure. What about using, for each $k = 1, \ldots K$:

$$\hat{p}_k^{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} 1\{\hat{f}^{\text{tree},b}(x) = k\}$$

I.e., the proportion of votes that were for class k?

This is generally not a good estimate. Simple example: suppose that the true probability of class 1 given x is 0.75. Suppose also that each of the bagged classifiers $\hat{f}^{\text{tree},b}(x)$ correctly predicts the class to be 1. Then $\hat{p}_1^{\text{bag}}(x) = 1$, which is wrong

What's nice about trees is that each tree already gives us a set of predicted class probabilities at x: $\hat{p}_k^{\text{tree},b}(x)$, $k = 1, \ldots K$. These are simply the proportion of points in the appropriate region that are in each class

Alternative form of bagging

This suggests an alternative method for bagging. Now given an input $x \in \mathbb{R}^p$, instead of simply taking the prediction $\hat{f}^{\text{tree},b}(x)$ from each tree, we go further and look at its predicted class probabilities $\hat{p}_k^{\text{tree},b}(x)$, $k = 1, \ldots K$. We then define the bagging estimates of class probabilities:

$$\hat{p}_k^{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{p}_k^{\text{tree},b}(x) \quad k = 1, \ldots K$$

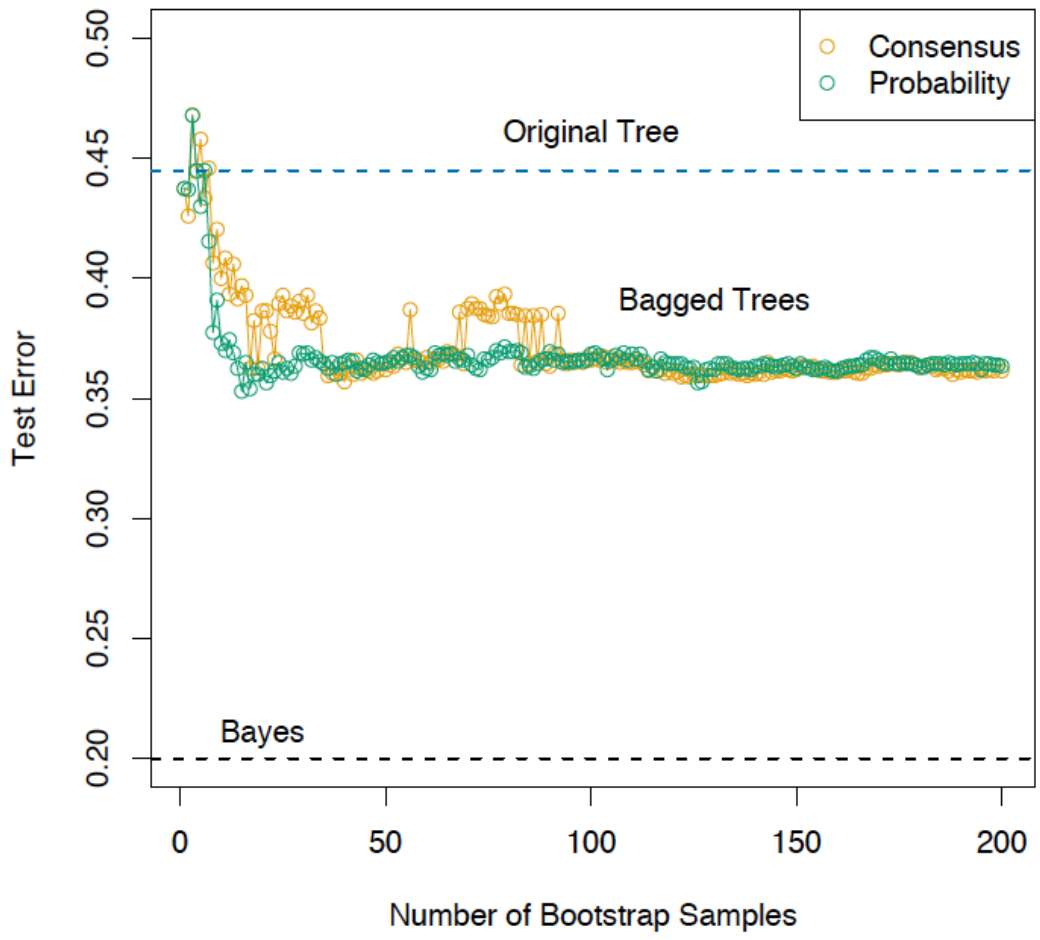
The final bagged classifier just chooses the class with the highest probability:

$$\hat{f}^{\text{bag}}(x) = \underset{k=1,\ldots K}{\text{argmax}} \ \hat{p}_k^{\text{bag}}(x)$$

This form of bagging is preferred if it is desired to get estimates of the class probabilities. Also, it can sometimes help the overall prediction accuracy

Example: alternative form of bagging

Previous example revisited: the alternative form of bagging produces misclassification errors shown in green



Why is bagging working?

Why is bagging working? Here is a simplified setup with $K = 2$ classes to help understand the basic phenomenon

Suppose that for a given input x, we have B independent classifiers $\hat{f}^b(x)$, $b = 1, \ldots B$, and each classifier has a misclassification rate of $e = 0.4$. Assume w.l.o.g. that the true class at x is 1, so

$$\mathrm{P}(\hat{f}^b(x) = 2) = 0.4$$

Now we form the bagged classifier:

$$\hat{f}^{\mathrm{bag}}(x) = \underset{k=1,2}{\mathrm{argmax}} \sum_{b=1}^{B} 1\{\hat{f}^b(x) = k\}$$

Let $B_2 = \sum_{b=1}^{B} 1\{\hat{f}^b(x) = 2\}$ be the number of votes for class 2

Notice that $\sum_{b=1}^{B} 1\{\hat{f}^b(x) = 2\}$ is a binomial random variable,

$$B_2 \sim \text{Binom}(B, 0.4)$$

Therefore the misclassification rate of the bagged classifier is

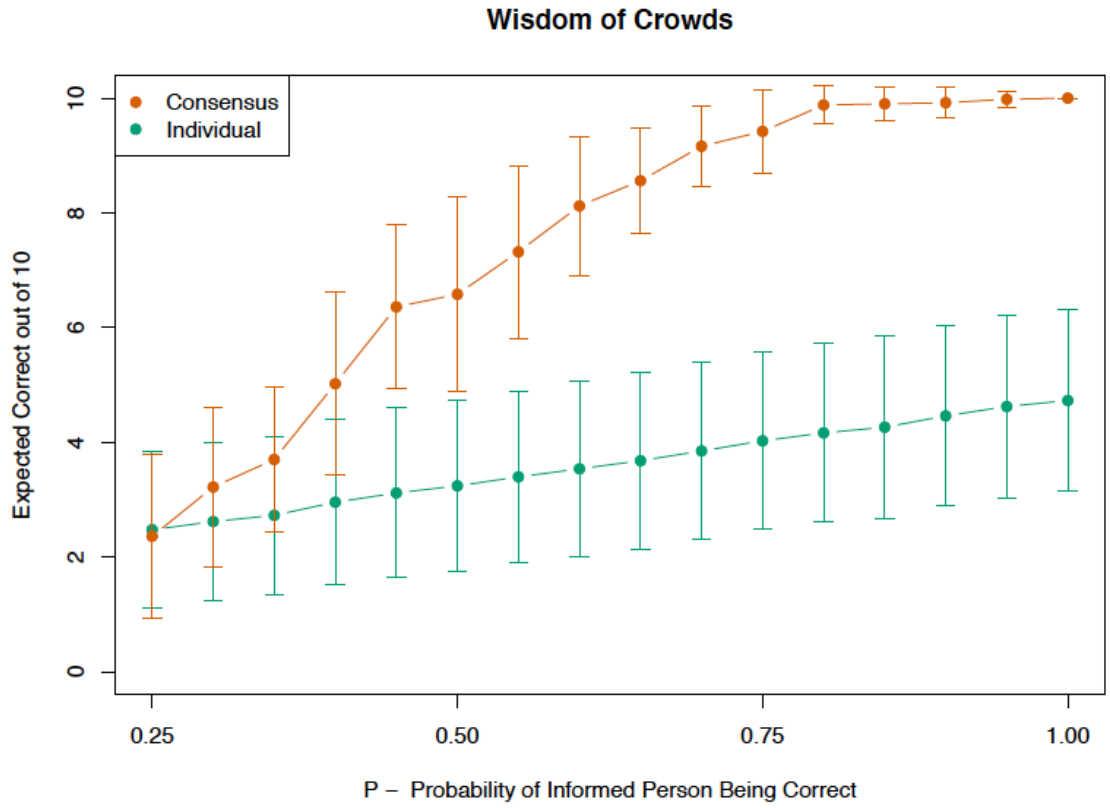
$$\text{P}(\hat{f}^{\text{bag}}(x) = 2) = \text{P}(B_2 \geq B/2)$$

As $B_2 \sim \text{Binom}(B, 0.4)$, this $\to 0$ as $B \to \infty$. In other words, the bagged classifier has perfect predictive accuracy as the number of sampled data sets $B \to \infty$

So why did the prediction error seem to level off in our examples? Of course, the caveat here is independence. The classifiers that we use in practice, $\hat{f}^{\text{tree},b}$, are clearly not independent, because they are fit on very similar data sets (bootstrap samples from the same training set)

Wisdom of crowds

The wisdom of crowds is a concept popularized outside of statistics to describe the same phenomenon. It is the idea that the collection of knowledge of an independent group of people can exceed the knowledge of any one person individually. Interesting example (from ESL page 287):



When will bagging fail?

Now suppose that we consider the same simplified setup as before (independent classifiers), but each classifier has a misclassification rate:

$$\mathrm{P}(\hat{f}^b(x) = 2) = 0.6$$

Then by the same arguments, the bagged classifier has a misclassification rate of

$$\mathrm{P}(\hat{f}^{\mathrm{bag}}(x) = 2) = \mathrm{P}(B_2 \geq B/2)$$

As $B_2 \sim \mathrm{Binom}(B, 0.6)$, this $\to 1$ as $B \to \infty$! In other words, the bagged classifier is perfectly inaccurate as the number of sampled data sets $B \to \infty$

Again, the independence assumption doesn't hold with trees, but the take-away message is clear: bagging a good classifier can improve predictive accuracy, but bagging a bad one can seriously degrade predictive accuracy

Disadvantages

It is important to discuss some disadvantages of bagging:

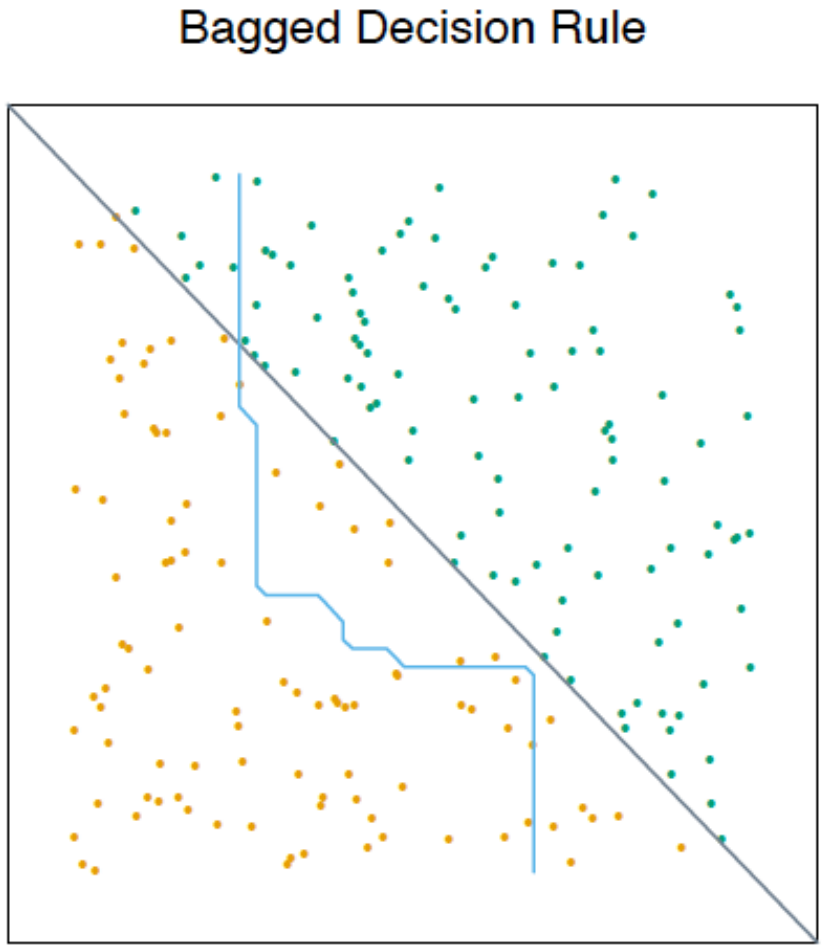
- *Loss of interpretability:* the final bagged classifier is not a tree, and so we forfeit the clear interpretative ability of a classification tree
- *Computational complexity:* we are essentially multiplying the work of growing a single tree by B (especially if we are using the more involved implementation that prunes and validates on the original training data)

You can think of bagging as extending the space of models. We go from fitting a single tree to a large group of trees. Note that the final prediction rule cannot always be represented by a single tree

Sometimes, this enlargement of the model space isn't enough, and we would benefit from an even greater enlargement

Example: limited model space

Example (from ESL page 288): bagging still can't really represent a diagonal decision rule



Recap: bagging

In this lecture we learned bagging, a technique in which we draw many bootstrap-sampled data sets from the original training data, train on each sampled data set individually, and then aggregate predictions at the end. We applied bagging to classification trees

There were two strategies for aggregate the predictions: taking the class with the majority vote (the consensus strategy), and averaging the estimated class probabilities and then voting (the probability strategy). The former does not give good estimated class probabilities; the latter does and can sometimes even improve prediction accuracy

Bagging works if the base classifier is not bad (in terms of its prediction error) to begin with. Bagging bad classifiers can degrade their performance even further. Bagging still can't represent some basic decision rules

Next time: boosting

Boosting is a powerful tool. In fact, Leo Breiman (the inventor of bagging!) once called it the "greatest off-the-shelf classifier in the world"