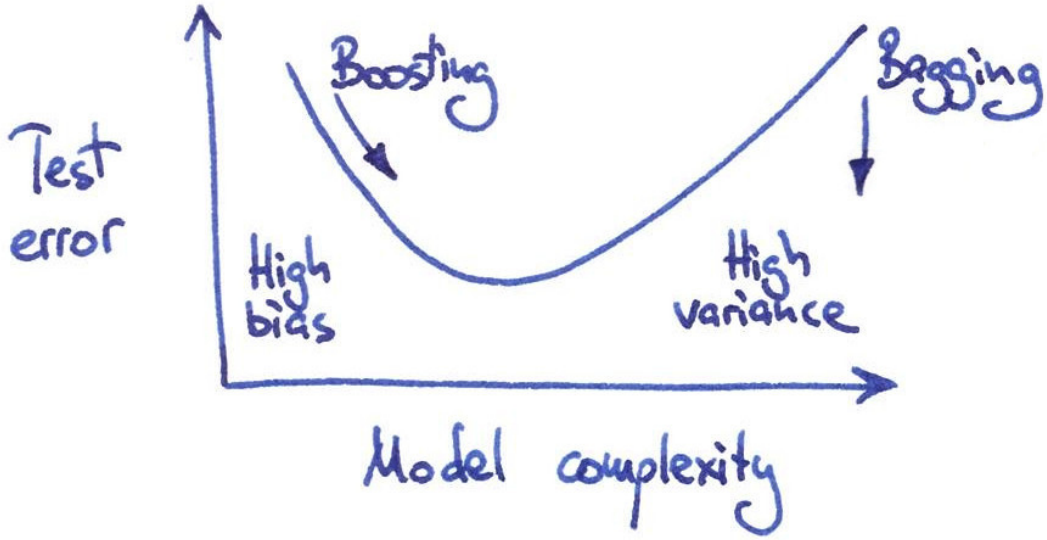
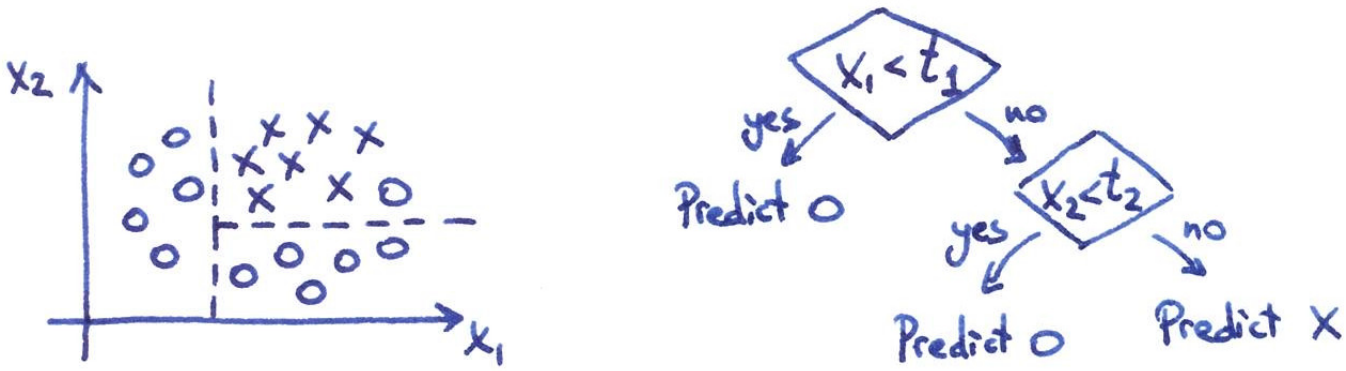
Boosting and bagging



- *Boosting* builds complex models out of simple ones (combats high bias).
- *Bagging* averages complex models to simplify them (combats high variance).

Classification trees

Both boosting and bagging often use binary *classification trees*:



The tree is built in a *greedy* fashion, iteratively looking for best splits.

The number of splits regulates model complexity, from a 'stump' (a single split) to a fully grown tree (100% training accuracy).

Note: trees can be used for regression as well (*regression trees*).

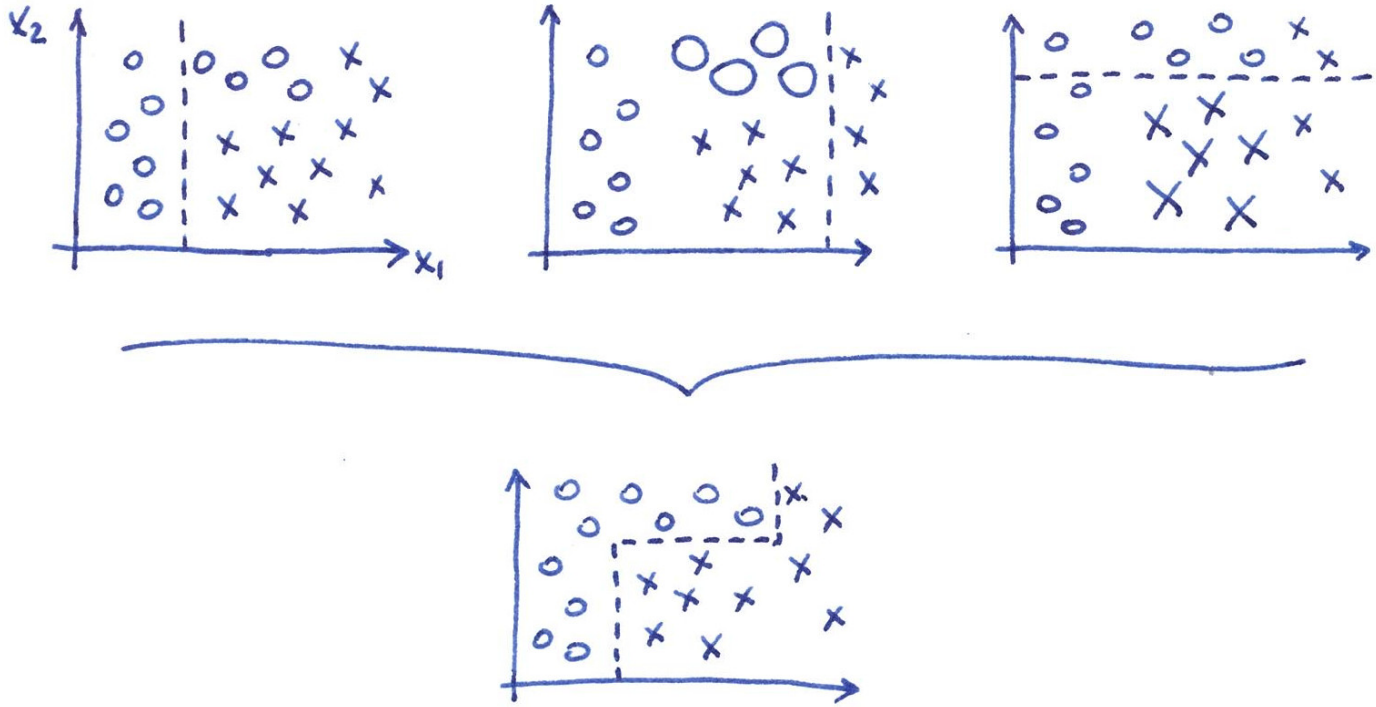
Boosting

Boosting repeatedly applies a weak binary classifier $G : \mathbb{R}^p \to \{-1, 1\}$ to the training set, modifying sample *weights*:

- Start with equal sample weights $w_i = 1/n$.
- For steps $m = 1 \ldots M$:
  - Fit a classifier G_m to the training data using weights w_i.
  - Compute its weight α_m given its performance.
  - Update the sample weights w_i to increase the importance of misclassified samples.
- Output $G(\mathbf{x}) = \text{sign}\left(\sum_m \alpha_m G_m(\mathbf{x})\right)$.

Boosting example

One can use binary tree stumps as the weak classifier.



AdaBoost

For each classifier G_m, define the weighted error rate as

$$\text{err}_m = \frac{\sum w_i I(y_i \neq G_m(\mathbf{x}_i))}{\sum_i w_i}.$$

AdaBoost (Freund & Shapire, 1997) sets $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$ and updates the weights as $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(\mathbf{x}_i))]$.

These formulas look pretty mysterious. Turns out (Friedman et al., 2000), that AdaBoost performs greedy optimization of the exponential loss function $\mathcal{L} = \sum_i \exp(-y_i G(\mathbf{x}_i))$ using an additive model $G(\mathbf{x}_i) = \sum_m \alpha_m G_m(\mathbf{x})$.

AdaBoost and the exponential loss

Using the exponential loss function and greedy optimization, on step m one optimizes

$$\sum_i \exp[-y_i(G_{m-1}(\mathbf{x}_i) + \alpha_m G_m(\mathbf{x}_i))],$$

where G_{m-1} denotes the model built on previous steps. This can be rewritten as

$$\sum_i w_i \exp[-\alpha_m y_i G_m(\mathbf{x}_i)].$$

From here one can show that G_m should minimize the weighted error rate, and derive AdaBoost's formulas for α_m and for w_i updates.

Note that this machinery works thanks to the exponential loss function. *Gradient boosting* is a powerful generalization to other loss functions.

Some comments on AdaBoost

- It is often considered one of the best off-the-shelf classifiers.
- Number of boosting iterations controls model complexity.
- AdaBoost can overfit in principle, but often overfits very slowly or does not overfit at all!
- Boosting long enough will increase the training accuracy to 100%.
- Exponential loss can keep decreasing even after the training accuracy is at 100% (and test accuracy can also keep increasing).

Bagging

Bagging (stands for 'boostrap aggregation') refers to model averaging of models constructed on *bootstrapped* datasets:

$$G(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} G_b(\mathbf{x}),$$

where each G_b is constructed on a bootstrapped dataset of size n.

Bootstrapping: randomly, with repetitions, select n samples out of n. This results in leaving out $\sim 1/e$ samples.

Bagging is applied to models with low bias and high variance. After averaging, bias remains low but variance decreases.

If different models were independent, then the variance would decrease to 0 when $B \to \infty$. But in reality the models are not independent.

Random Forests

Random Forests (Breiman, 2001) use bagging of fully grown trees, but with additional randomness introduced in order to decrease the dependence:

- for each split, randomly select $m \ll p$ variables as candidates for splitting. By default, $m = \sqrt{p}$.

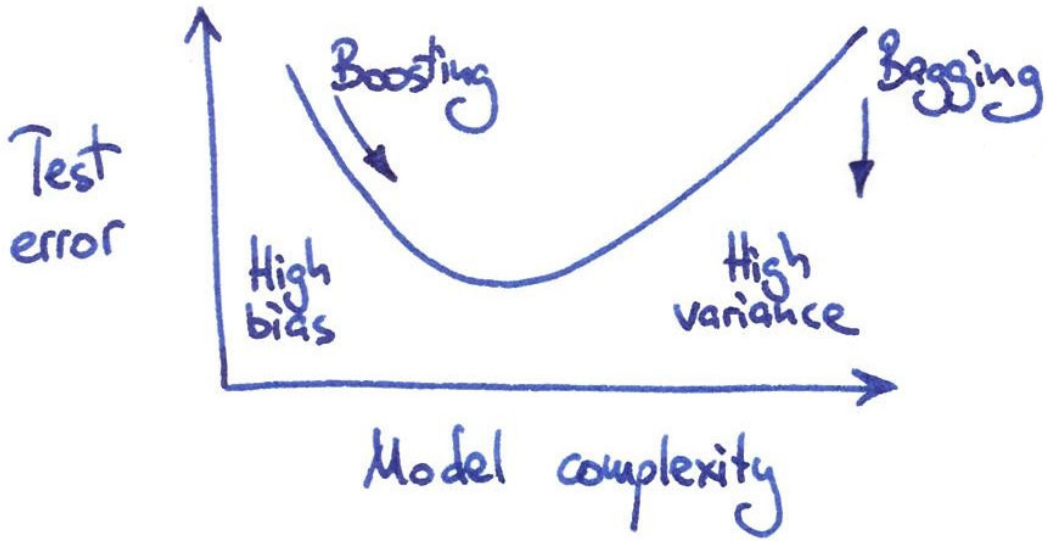
Note that here — unlike in boosting — the number of trees (B) does not regulate model complexity.

Some comments on Random Forests

- They are also often considered one of the best off-the-shelf classifiers.
- A random forest often performs similarly to AdaBoost / gradient boosting.
- It requires very little tuning.
- It shows 100% training set accuracy.
- It does not need cross-validation or a test set: one can test the performance on the 'out-of-bag' samples. For each sample i, average trees constructed on boostrapped datasets that did not include i.
- It allows easy assessment of variable importance: when checking the performance on the out-of-bag samples, randomly permute each variable one after another. The average decrease in accuracy quantifies each variable's importance.

AdaBoost and Random Forests

Both are *interpolating* classifiers, i.e. they have perfect training accuracy.



See e.g. Wyner et al. (2017). Also see Lectures 4 and 7.