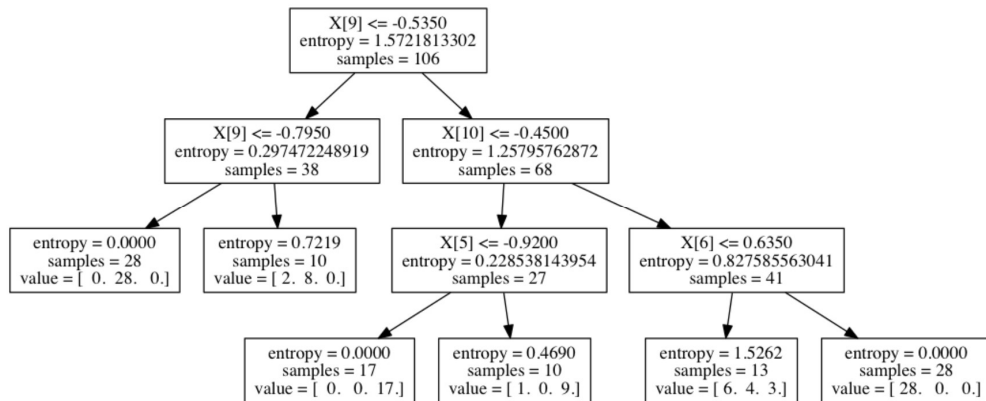
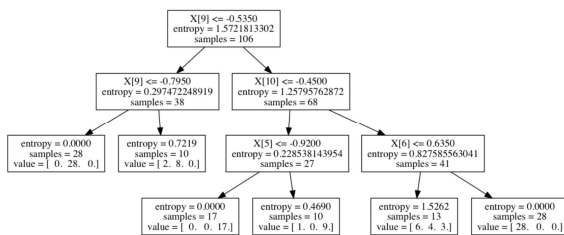


## Decision trees

$$G(D, x) = \text{Entropy}(D) - \sum_{v \in x} \frac{|D_v|}{|D|} \text{Entropy}(D_v)$$



## Decision trees



advantages:

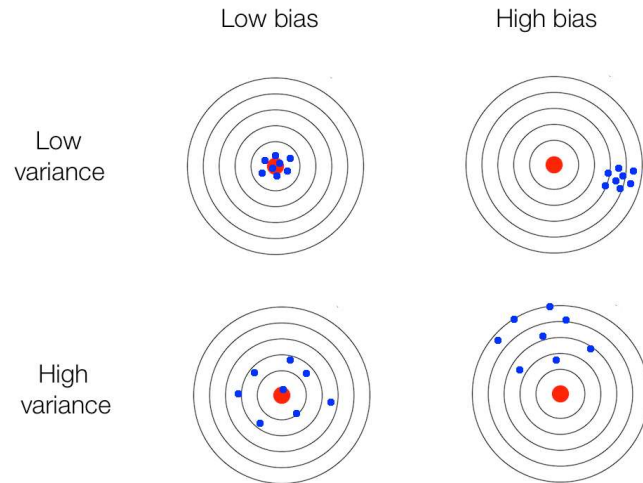
- ease of interpretation
- handles continuous and discrete features
- invariant to monotone transformation of features
- variable selection automated
- **low bias** (deep trees)

disadvantages:

- **high variance**
- overfitting

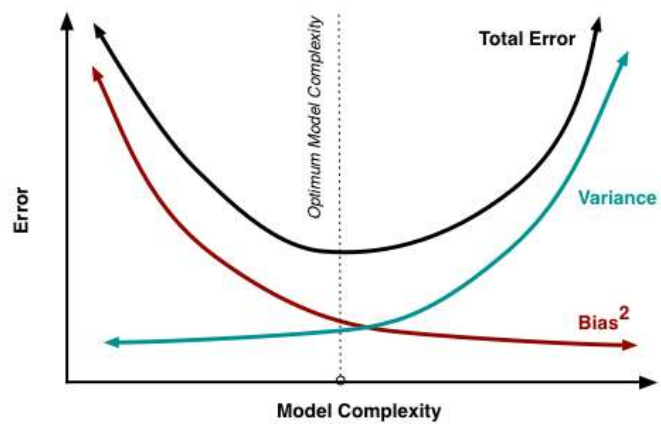
### Bias-variance tradeoff

- Error due to bias
- Error due to variance
- Irreducible error

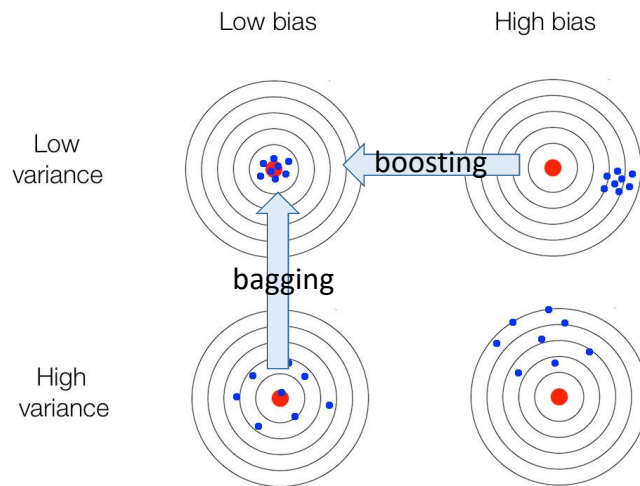


### Bias-variance tradeoff

- Error due to bias
- Error due to variance
- Irreducible error



## *bagging/boosting*



## *Bagging*

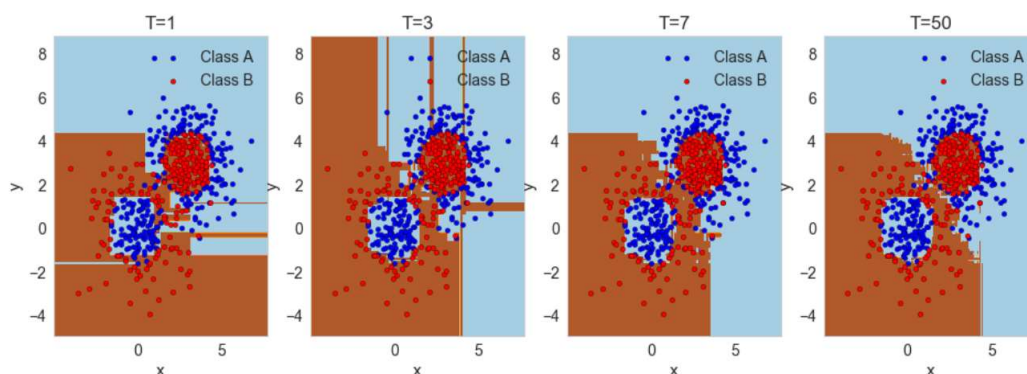
- Simulate the notion of different train set samples:
  1. sample data points from train set to create new train set
  2. fit low bias high variance model on new train set
  3. repeat steps (1) and (2)  $T$  times
  4. average the predictions of the  $T$  models

$$\hat{f}(x, \theta) = \frac{1}{T} \sum_{t=1}^T f_t(x, \theta)$$

## *Bagging: Random Forests*

- Train set contains  $n$  data points with  $m$  features.
- Construct  $T$  low bias high variance decision trees by following these steps:
  1. Sample  $n$  data points at random **with replacement** from the train set.
  2. At each node, select  $h \ll m$  features at random and compute the best split using only these  $h$  features.
  3. Each tree is grown to the largest extent possible. There is no pruning or early stopping.
- Step 3 ensures that the bagged models are low bias by learning deep complex decision trees.

## *Bagging: Random Forests*



## *Boosting*

- reduce the bias of a high bias low variance model
- turning an ensemble of weak learners into a strong learner
- the meta-model is additive:

$$\hat{f}(x, \theta) = \sum_{t=1}^T \alpha_t f_t(x, \theta)$$

## *Boosting: adaboost*

Initialize the weights  $D_1(i) = 1/n, i = 1, 2, \dots, n$ .

For  $t = 1$  to  $T$ :

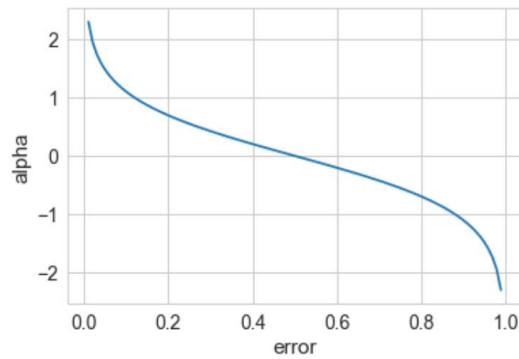
1. Fit a weak classifier  $f_t(x, \theta)$  to the trainset data using weights  $D_1(i)$ .
2. Set  $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\text{error}}{\text{error}}\right)$ .
3. Update weights:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i f_t(x_i, \theta))}{Z_t},$$

where  $Z_t$  is a normalizing factor that makes sure that  $\sum D_{t+1}(i) = 1$ .

$$\hat{f}(x, \theta) = \sum_{t=1}^T \alpha_t f_t(x, \theta)$$

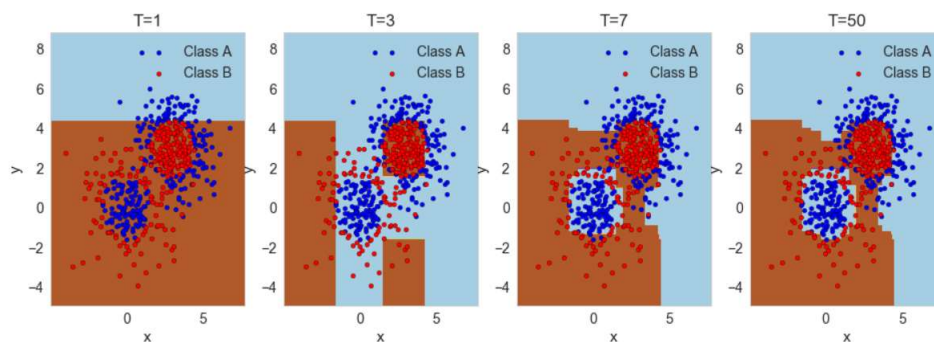
## Boosting: adaboost



- The weight of a weak model in the boosted meta-model increases exponentially as the error approaches 0. Better models are given exponentially more weight.
- The weight is zero if the error rate is 0.5. A model with 50% accuracy is no better than random guessing, so it is ignored.
- The weight decreases exponentially as the error approaches 1. A negative weight is given to classifiers with worse than 50% accuracy. "Whatever that classifier says, do the opposite!"

$$\hat{f}(x, \theta) = \sum_{t=1}^T \alpha_t f_t(x, \theta)$$

## Boosting: adaboost



### *Boosting: Gradient Boosting*

$$\hat{f}(x, \theta) = \sum_{t=1}^T f_t(x, \theta)$$

1. Fit a model  $f_1(x, \theta) = y$

2. Fit a model to the **residuals**  $h_1(x) = y - f_1(x, \theta)$

3. Create a new model  $f_2(x, \theta) = f_1(x, \theta) + h_1(x)$

$$f_0(x, \theta) = \frac{1}{n} \sum_{i=1}^n y_i$$

$$f_{t+1}(x, \theta) = f_t(x, \theta) + h_t(x)$$