CS 559 Machine Learning

Decision Trees

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Plan for today

Decision Tree Learning

Ensemble Methods

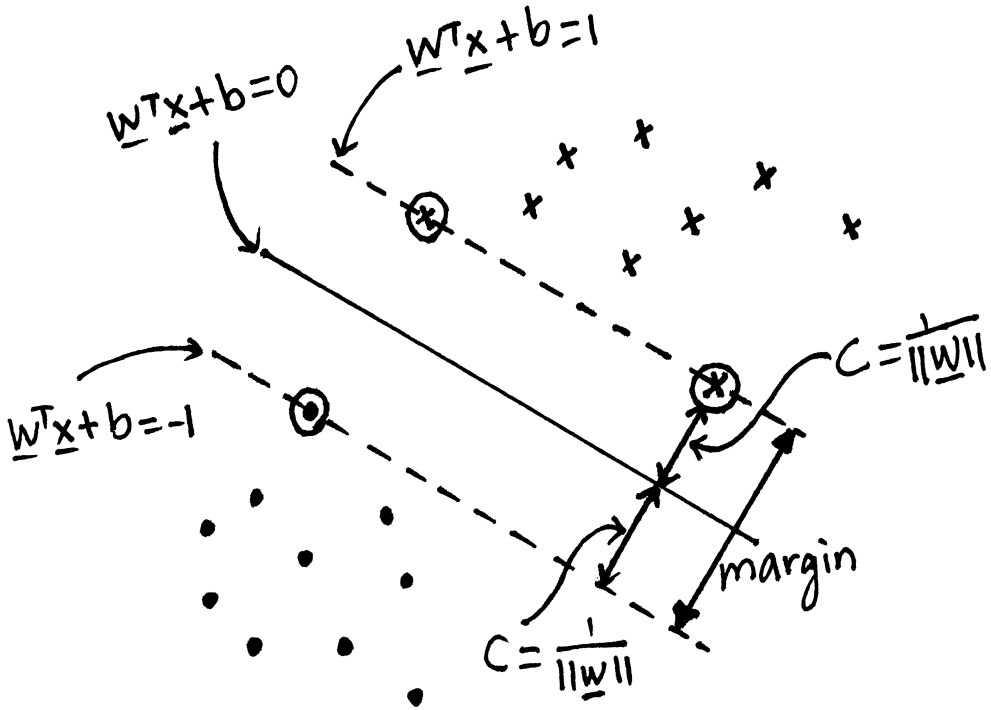
Random Forests

Learning Outcomes

- Explain decision tree (entropy, information gain, split, when to stop)
- Explain reasons for using ensemble and types of ensemble
- Explain random forest

Review of last lecture -SVM

Margin: the smallest distance between the decision boundary and any of the training samples.



Review of last lecture - SVM formulation

- Primal form:

$$\min_{\mathbf{w},b} L_p = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

- Dual form:

$$\max L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k$$

$$\text{subject to } \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right] = 0 \quad \forall i = 1, ..., N$$

Review of last lecture - SVM dual form

- The solution vector $\mathbf{w}$ is: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$.
- Support vectors (those $\mathbf{x}_i$ for which $\alpha_i > 0$)
- To obtain the value of b: solve $\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right] = 0$ for any of the support vectors.
- The largest margin hyperplane gives a function: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$ for classifying new observations $\hat{y} = \text{sign}(f(\mathbf{x}))$

Decision boundary, Loss function, Gradients

- Decision boundary:
  - $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$: linear discriminant function
  - $f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) = 0.5$: logistic regression
- Loss (error, objective) function on one example:
  - $L(\mathbf{x}, y, \mathbf{w}) = (f(\mathbf{x}) - y)^2$: squared error
  - $L(\mathbf{x}, y, \mathbf{w}) = -[y_i \log f(\mathbf{x}) + (1 - y_i) \log(1 - f(\mathbf{x}))]$: negative log-likelihood
  - $L(\mathbf{x}, y, \mathbf{w}) = \max(0, 1 - y \cdot f(\mathbf{x}))$: hinge loss in SVM
- Gradient (derivative): take (partial) derivative of a loss function with respect to your model parameters ($\mathbf{w}$)
  - notation: $\nabla_{\mathbf{w}} L$
  - notation: $g(\mathbf{w})$
  - notation: $d\mathbf{w}$
  - notation: $\frac{\partial L}{\partial \mathbf{w}}$

A learning problem: predict fuel efficiency

Table: From the UCI repository

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

- 40 data points
- Goal: predict MPG (good or bad)
- Need to find: $f : X \to Y$
- Discrete features/attributes

Hypotheses: decision trees $f : X \rightarrow Y$

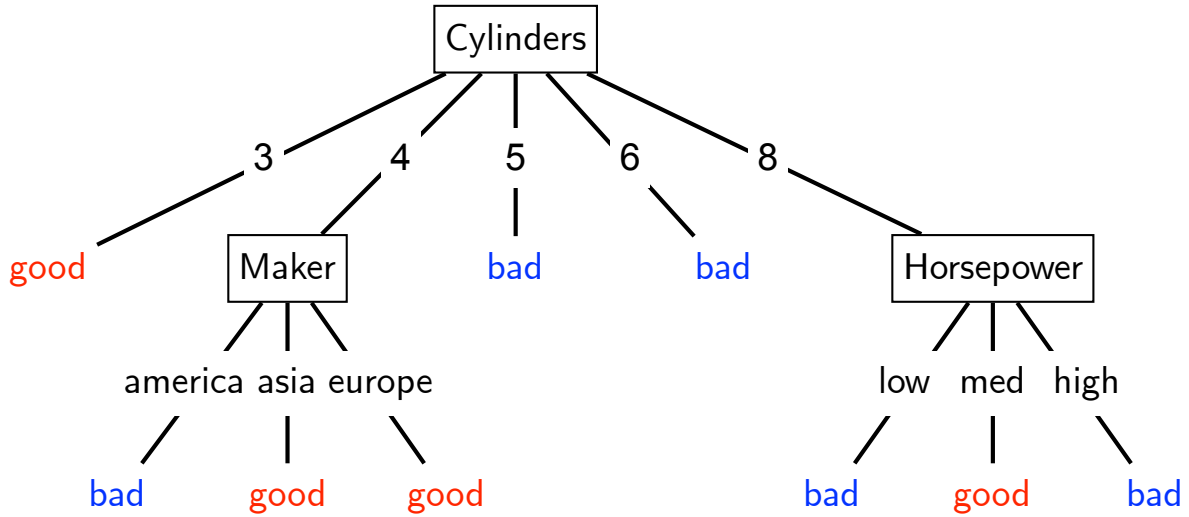


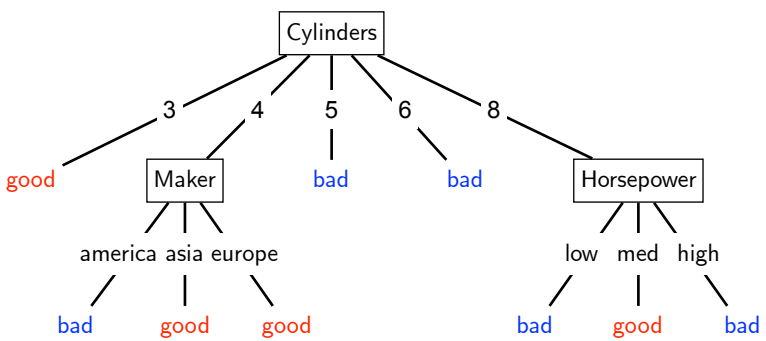
Figure: Human interpretable!

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class
- To classify input **x**: traverse the tree from root to leaf, output the labeled y

Hypothesis space

Table: From the UCI repository

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

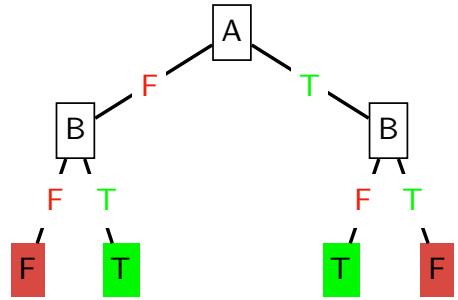


- How many possible hypotheses?
- What functions can be represented?

What functions can be represented?

- Decision trees can represent any function of the input attributes
- For Boolean functions, one path to leaf gives a truth table row
- But, it could require exponentially many nodes...

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F



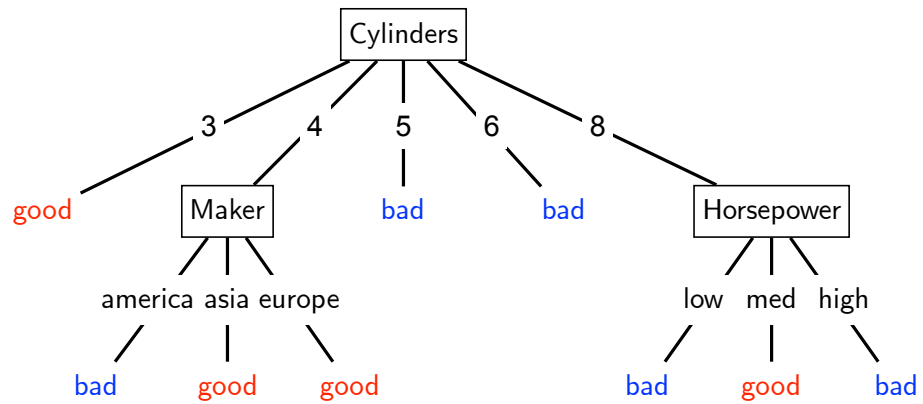


Figure: Predicting mpg=good: $\text{cyl} = 3 \vee (\text{cyl} = 4 \wedge (\text{maker} = \text{asia} \vee \text{maker} = \text{europe})) \vee \ldots$

Hypothesis space

- How many possible hypotheses?
- What functions can be represented?
- How many will be consistent with a given dataset?
- How will we choose the best one?
  - First look at how to split nodes, then consider how to find the best tree.

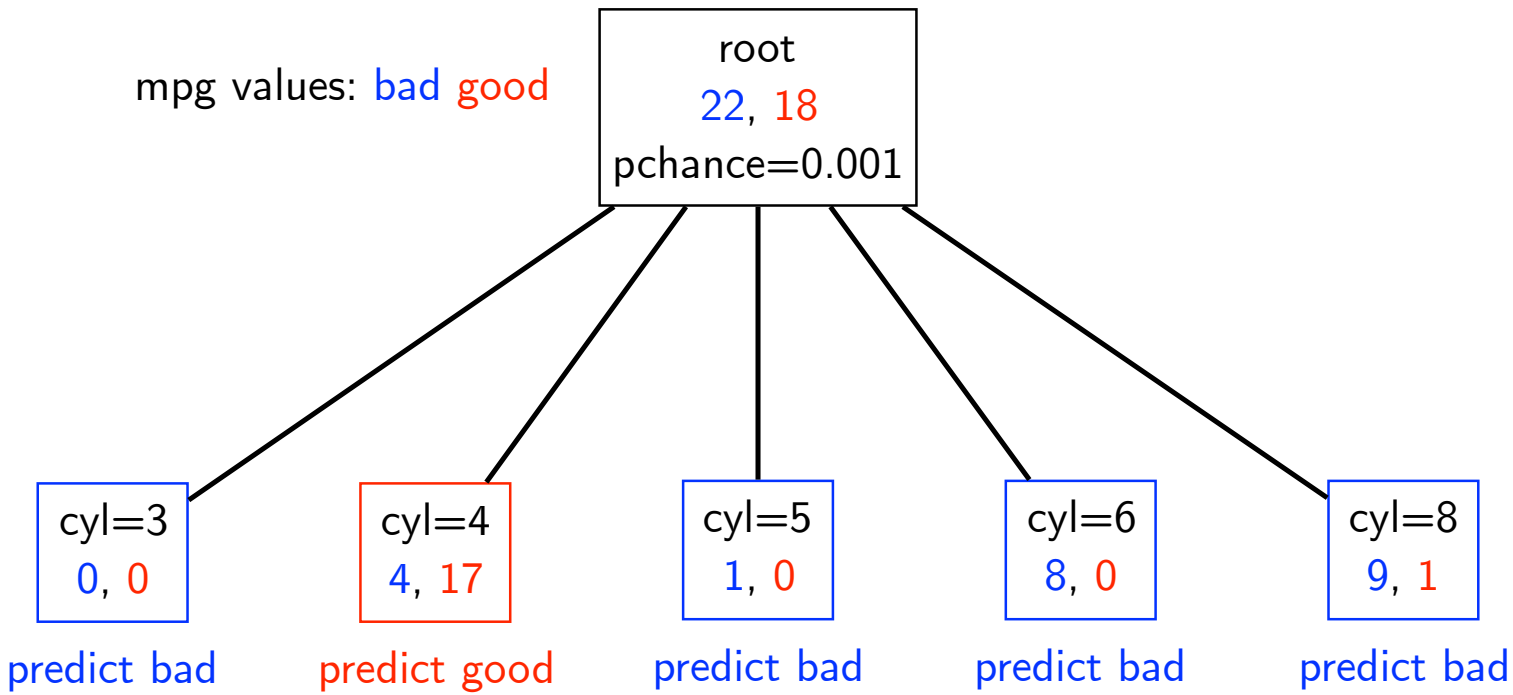
What is the simplest tree?

Table: From the UCI repository

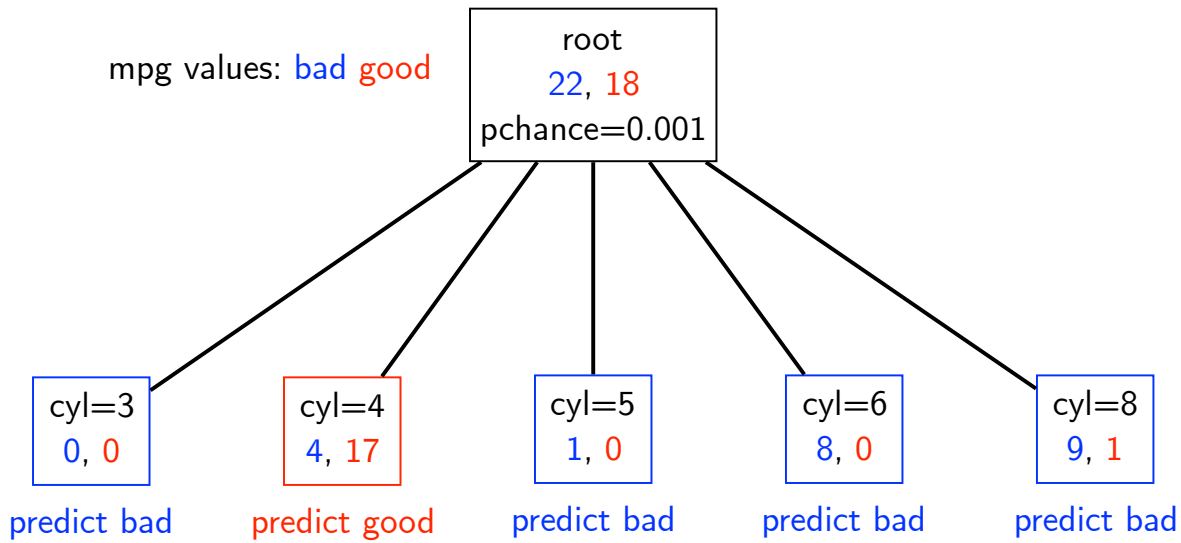
cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

- Predict mpg=bad
- Is this a good tree? Total we get $(22+, 18-)$, which means we are correct on 22 examples and wrong on 18 examples.

A simple decision tree

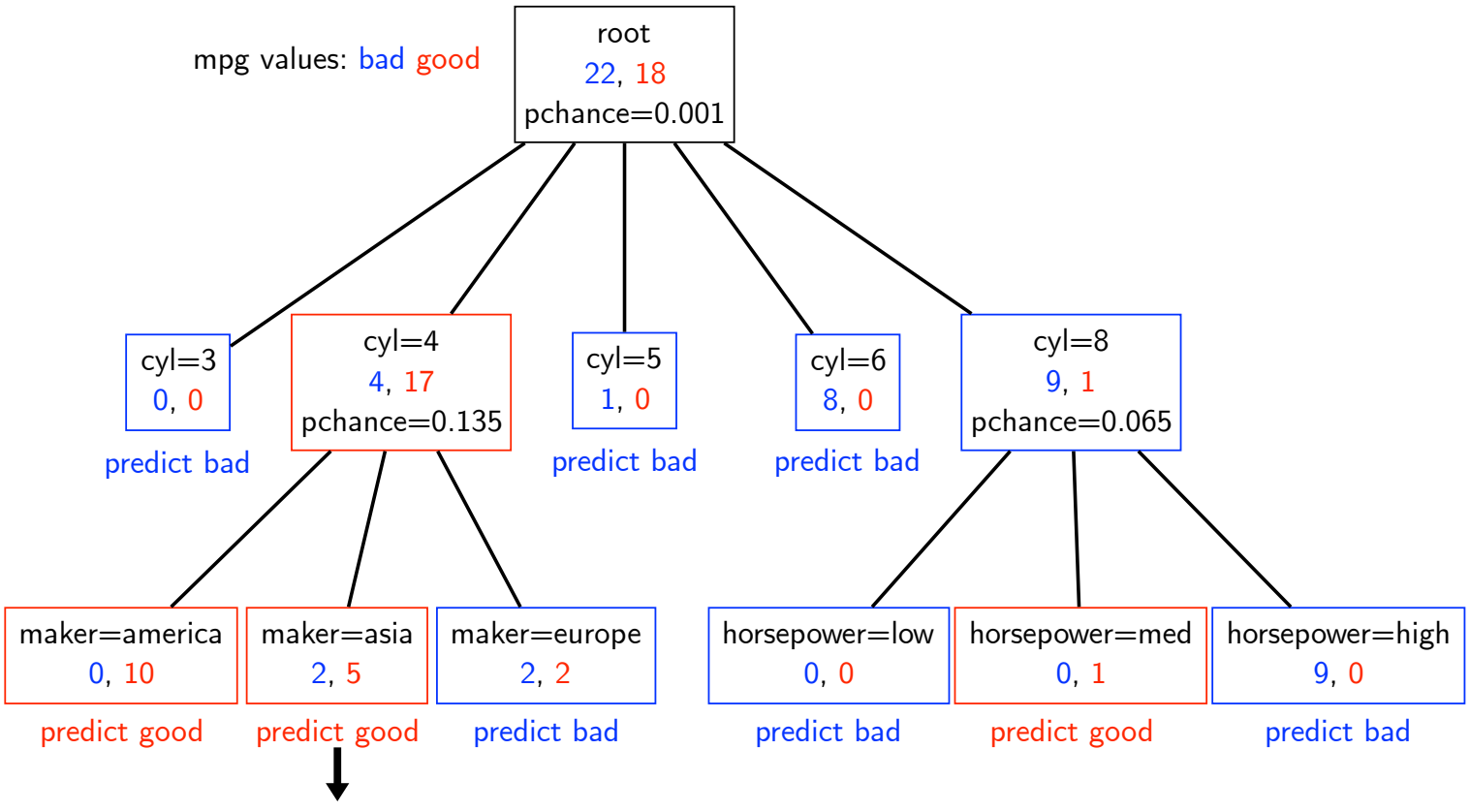


Recursive step

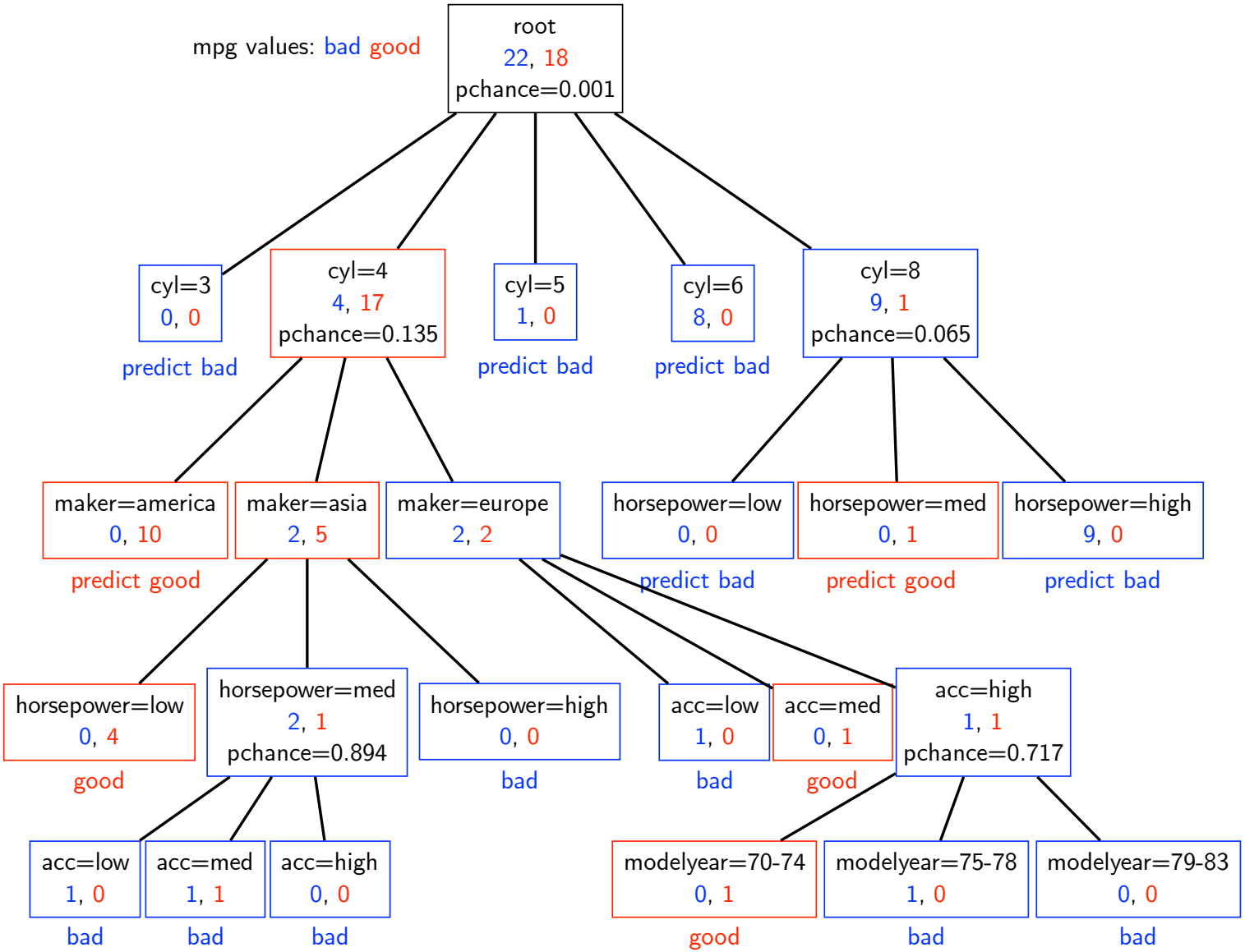


- Take the original dataset
- Partition it according to the values of the attribute we split on
- Build tree from these records (cyl=4, cyl=5, cyl=6, cyl=8)

Second level of a decision tree

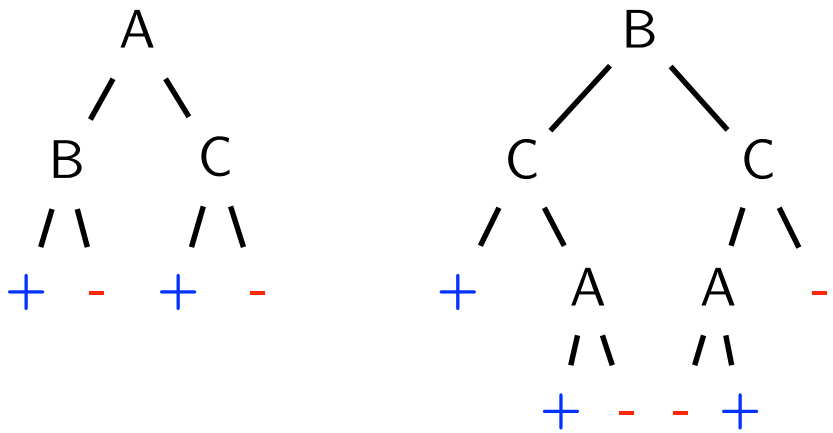


A full decision tree



Are all decision trees equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!



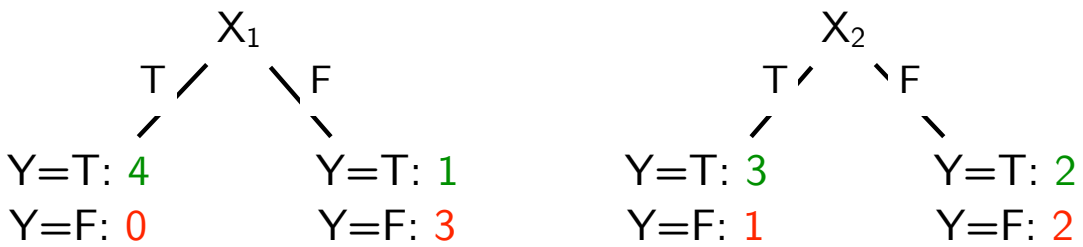
- Which tree do we prefer?

Learning decision trees is hard

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest 76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recurse

Splitting: choosing a good attribute

- Would we prefer to split on X_1 or X_2?



X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

- Idea: use counts as leaves to define probability distribution, so we can measure uncertainty.

Splitting: choosing a good attribute

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad
  - What about distributions in between?

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8
P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

Entropy

- Entropy $H(Y)$ of a random variable Y:

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

- More uncertainty, more entropy!
- Information theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code) Youtube video

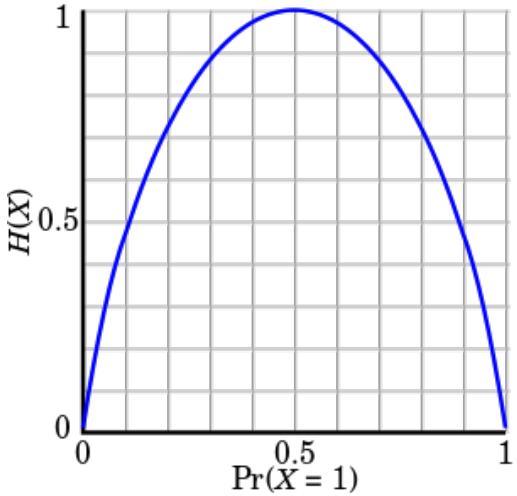


Figure: Entropy of a coin flip

Entropy

- High Entropy
  - Y is from a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- Low Entropy
  - Y is from a varied distribution (peaks and valleys)
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

Entropy example

Entropy:

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

In this example:

$$\begin{aligned} P(Y = T) &= 5/6 \\ P(Y = F) &= 1/6 \\ H(Y) &= -5/6 \log 5/6 - 1/6 \log 1/6 \\ &= 0.65 \end{aligned}$$

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional entropy

Conditional entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X:

$$H(Y|X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{K} P(Y = y_i|X = x_j) \log P(Y = y_i|X = x_j)$$

In this example:

$$P(X_1 = T) = 4/6$$
$$P(X_1 = F) = 2/6$$
$$H(Y|X_1) = -4/6(1 \log 1 + 0 \log 0) - 2/6(1/2 \log 1/2 + 1/2 \log 1/2) = 2/6$$

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Information gain

Used by the ID3, C4.5 and C5.0 tree-generation algorithms.
Decrease in entropy (uncertainty) after splitting:

$$IG(X) = H(Y) - H(Y|X)$$

In this example:

$$\begin{aligned} IG(X_1) &= H(Y) - H(Y|X_1) \\ &= 0.65 - 0.33 \end{aligned}$$

We prefer the split ($IG(X_1) > 0$)

X_1	X_2	**Y**
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

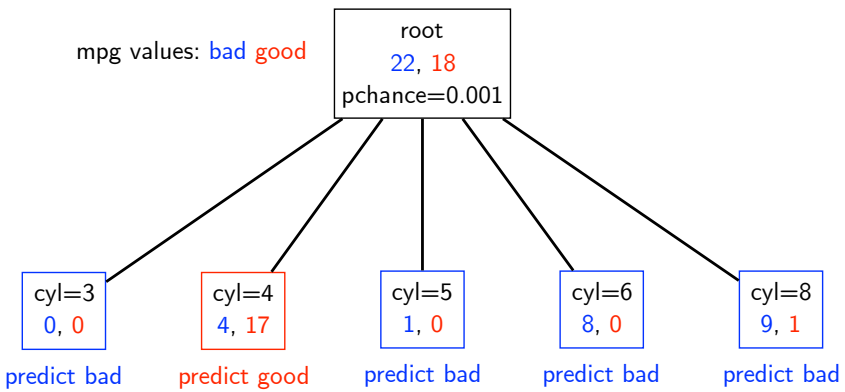
Learning decision trees

- Start from empty decision tree
- Split on next best attribute (feature) – Use information gain to select attribute
- Recurse

Information Gains - example

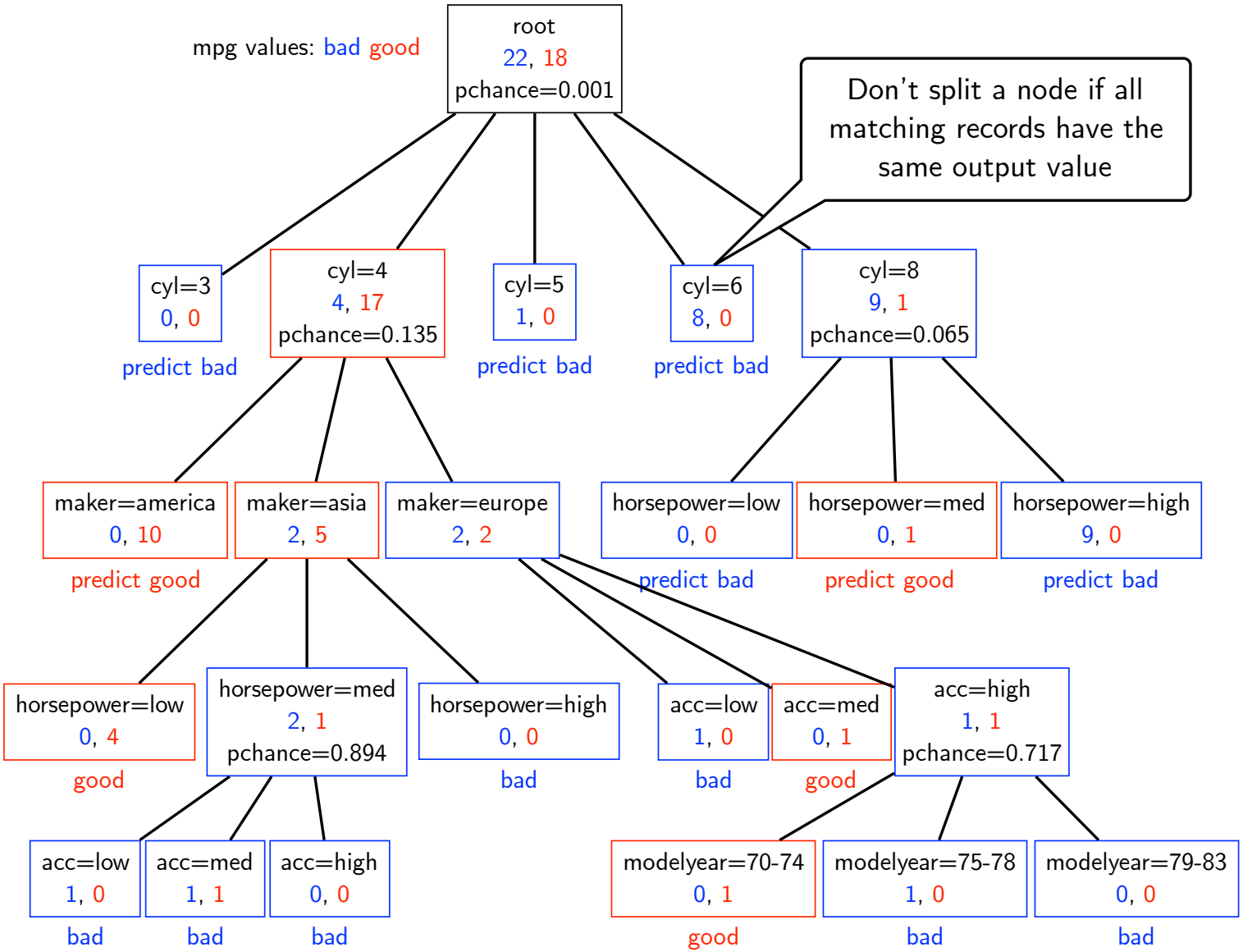
Table: Information gains using the training data set (40 records)

Input	value	Info Gain
cylinders	(3,4,5,6,8)	0.507
displacement	(low, medium, high)	0.223
horsepower	(low, medium, high)	0.388
weight	(low, medium, high)	0.304
acceleration	(low, medium, high)	0.064
modelyear	(70-74, 75-78, 79-83)	0.268

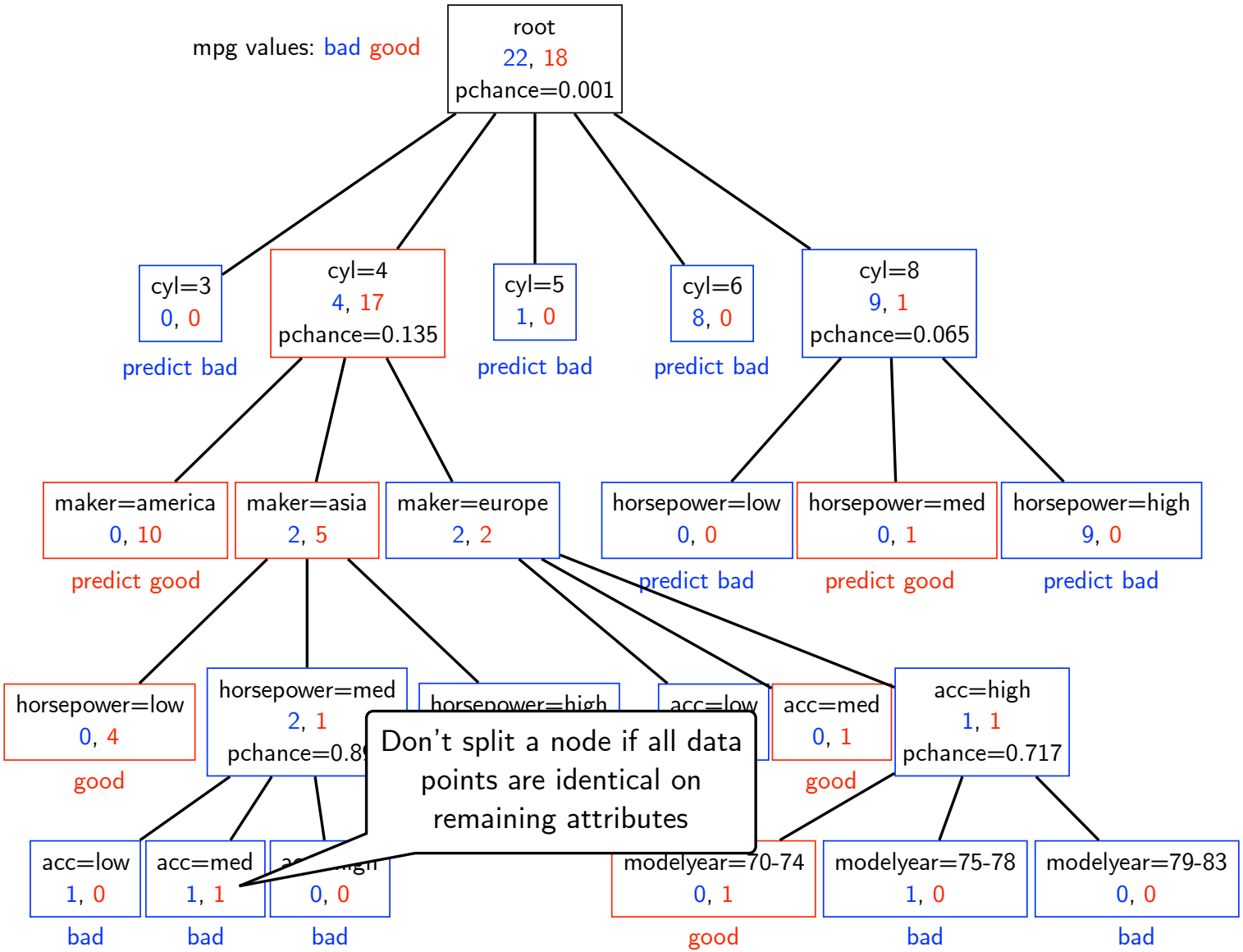


First split decided (cylinder), but when do we stop?

Base case one



Base case one



Base cases: an idea

- Base case one: if all records in current data subset have the same output then don't recurse
- Base case two: if all records have exactly the same set of input attributes then don't recurse.
- Proposed base case 3: If all attributes have zero information gain then don't recurse (is this a good idea?)

The problem with base case 3

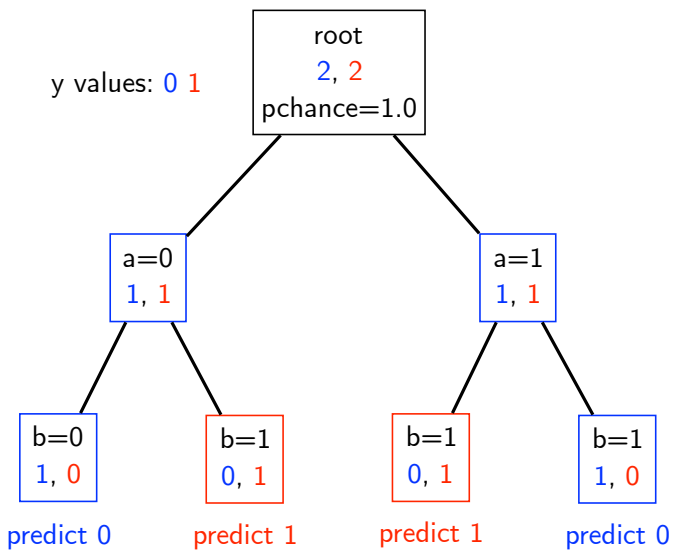
$y = a$ XOR b

a	b	**y**
0	0	0
0	1	1
1	0	1
1	1	0

$$IG(a) = 0$$
$$IG(b) = 0$$

If we omit base case 3

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



Other Metrics

- Gini index, used by the CART (classification and regression tree) algorithm, the perfect score is 0:
  - $I_G(p) = 1 - \sum_{j=1}^{K} p_j^2$, let p_j be the fraction of items labeled with class j in the set.
  - a perfect separation results in a Gini score of 0

Summary: Building decision trees

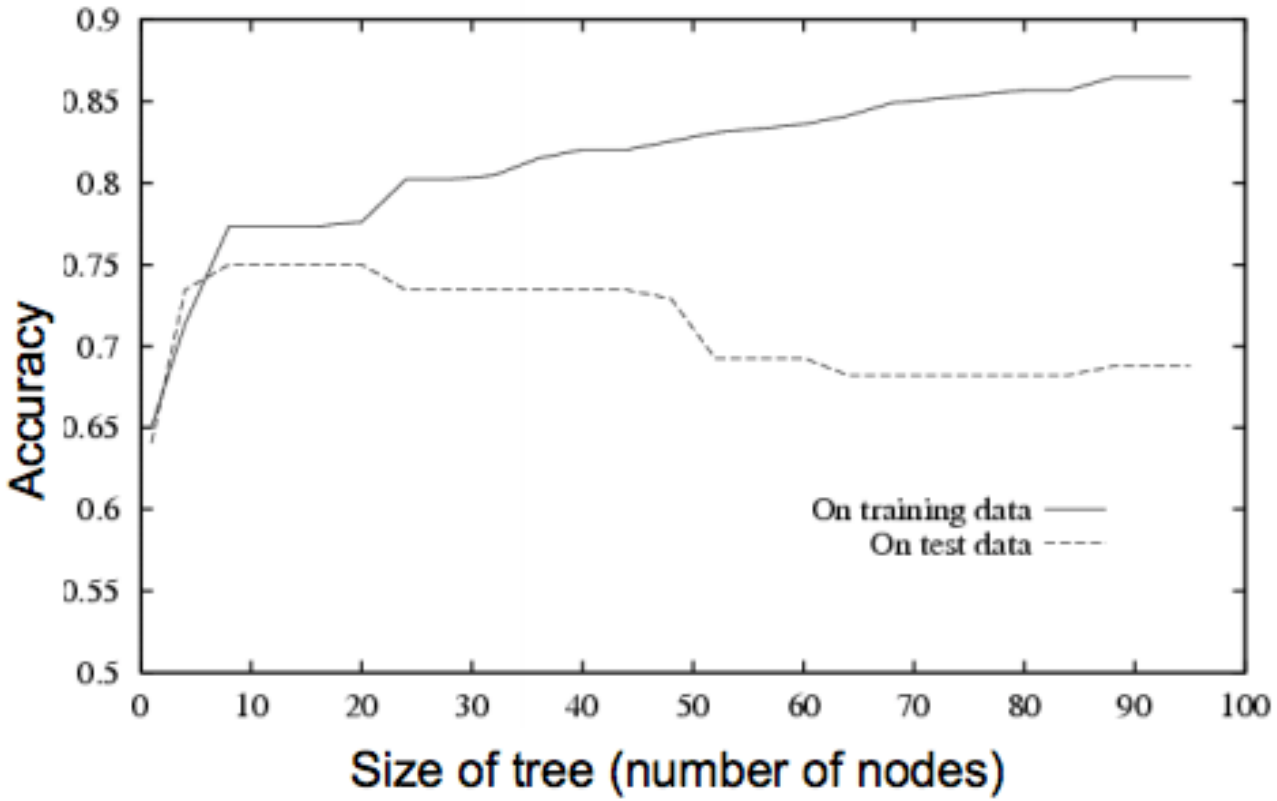
BuildTree(dataset, output)

- If all output values are the same in the dataset, return a leaf node that say predict this unique output.
- If all input values are the same, return a leaf node that says "predict the majority output"
- Else find attribute X with highest Info Gain (or lowest Gini Index)
- Suppose X has n_x distinct values
  - Create a non-leaf node with n_x children
  - The i-th child should be built by calling BuildTree(DS_i, output) where DS_i contains the records in dataset where $X = i$th value of X.

Decision trees overfit

- Standard decision trees have no learning bias.
  - Training set error is almost zero
  - Lots of variance
  - Must introduce some bias towards simpler trees
- Many strategies for picking simpler trees
  - Fixed depth
  - Fixed number of leaves
  - Or something smarter

Decision trees overfit



How to build small trees

Two reasonable approaches

- Optimize on the held-out (validation) set
  - If growing the tree larger hurts performance, then stop growing
  - Requires a large amount of data
- Use statistical significance testing
  - Test if the improvement for any split is likely due to noise. If so, don't do the split
  - Prune the tree bottom-up. A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance. Delete splits in which $p_{\text{chance}} > \max_{\text{chance}}$

Real-valued inputs

What should we do if some of the inputs are real-valued?

- Infinite number of possible split values
- Finite dataset, only finite number of relevant splits.

Proposed solution:

- One branch for each numeric value?
- Hopeless: hypothesis with such a high branching factor will shatter any dataset and overfit.

Threshold splits

- Binary tree: spilt on attribute X at value t:
  - One branch: $X < t$
  - Other branch: $X \geq t$
- Requires small change:
  - Allow repeated splits on same variable
  - How does this compare to "branch on each value" approach?

The set of possible thresholds

- Binary tree: spilt on attribute X at value t:
  - One branch: $X < t$
  - Other branch: $X \geq t$
- Search through possible values of t (seems hard!!!)
- But only a finite number of t's are important:
  - Sort data according to X into $\{x_1, ..., x_m\}$
  - Consider split points of the form $x_i + \frac{x_{i+1} - x_i}{2}$
  - Moreover, only splits between examples of different classes matter.

Picking the best threshold

- Suppose X is real valued with threshold t:
- Want $IG(Y|X:t)$, the information gain for Y when testing if X is greater than or less than t
- Define
  - $H(Y|X:t) = P(X < t)H(Y|X < t) + P(X \geq t)H(Y|X \geq t)$
  - $IG(Y|X:t) = H(Y) - H(Y|X:t)$
  - $IG^*(Y|X) = \max_t IG(Y|X:t)$
- Use $IG^*(Y|X)$ for continuous variables.

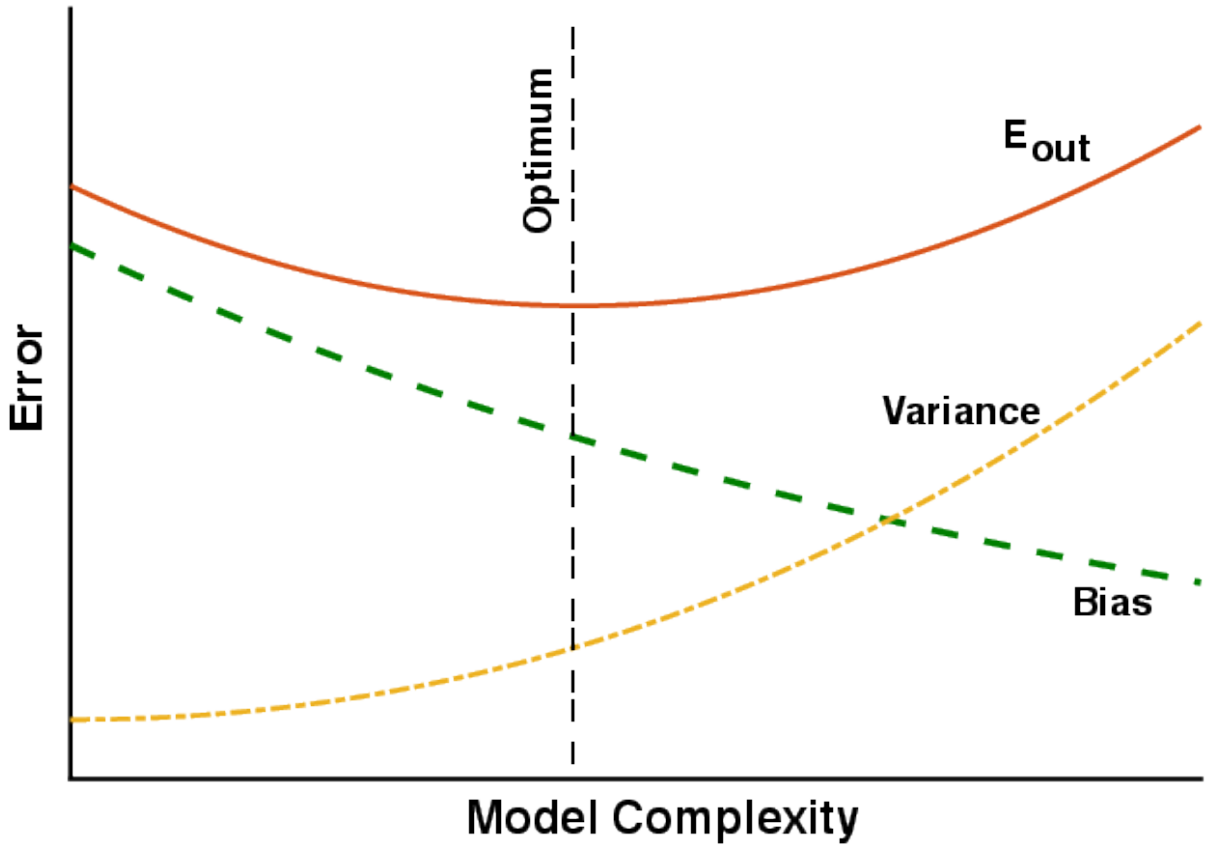
Summary

- Decision trees are one of the most popular ML tools
  - Easy to understand, implement, and use
  - Computationally cheap (to solve heuristically)
- Select attributes (ID3, C4.5, CART)
- Presented for classification, can be used for regression and density estimation too
- Decision trees will overfit
  - Must use tricks to find "simple trees"
  - Fixed depth/early stopping, pruning, hypothesis testing

Decision Trees vs. Linear Regression

- Decision trees
  - Can solve both classification and regression problem
  - If the relationship is non-linear it will outperform Linear regression
  - Can build models that are easy to explain
- Linear regression
  - In a linear relationship, linear regression will likely outperform Decision trees.
  - Cannot easily handle categorial variables.

Bias/Variance Tradeoff



Review: Bias and Variance

Expected squared prediction error at a point x is:

$$E[(y - \hat{f}(x))^2] = (E[\hat{f}(x) - f(x)])^2 + (E[\hat{f}(x)^2] - E^2[\hat{f}(x)]) + \sigma^2$$
$$= \text{Bias}^2 + \text{Variance} + \sigma^2$$

where

$$\text{Bias}[\hat{f}(x)] = E[\hat{f}(x) - f(x)], \text{Var}[\hat{f}(x)] = E[\hat{f}(x)^2] - E^2[\hat{f}(x)]$$

- Both contribute to errors
- Bias: error from incorrect modeling assumption
- Variance: error from random noise

Reduce variance without increasing bias

- Averaging reduces variance (when predictions are independent):

$$Var(\bar{X}) = \frac{Var(X)}{N}$$

- Average models to reduce model variance
  - In any network, the bias can be reduced at the cost of increased variance
  - In a group of networks, the variance can be reduced at no cost to bias
- One problem: only one training set, where do multiple models come from?

Ensemble Classifiers

- Basic idea: build different "experts" and let them vote
- Advantages:
  - Improve predictive performance
  - Different types of classifiers can be directly included
  - Easy to implement
  - Not too much parameter tuning
- Disadvantages:
  - The combined classifier is not transparent (black box)
  - Not a compact representation

Ensemble Methods

Predict class label for unseen data by aggregating a set of predictions (classifiers learned from the training data)

- Bagging (Breiman 1994 "Bagging Predictors")
- Random forests (Breiman 2001 "Random Forests")
- Boosting (Freund and Schapire 1995, Friedman et al. 1998,)

Reasons for using Ensembles

- Large volumes of data: Sometimes, the amount of data to be analyzed can be too large to be handled by a single classifier.
  - Partition the data into smaller subsets;
  - Train different classifiers;
  - Combine their outputs using a combination rule

Reasons for using Ensembles

- Too little data: A reasonable sized set of training data is crucial to learn the underlying data distribution.
  - Draw overlapping random subsets of the available data using resampling techniques
  - Train different classifiers, creating the ensemble

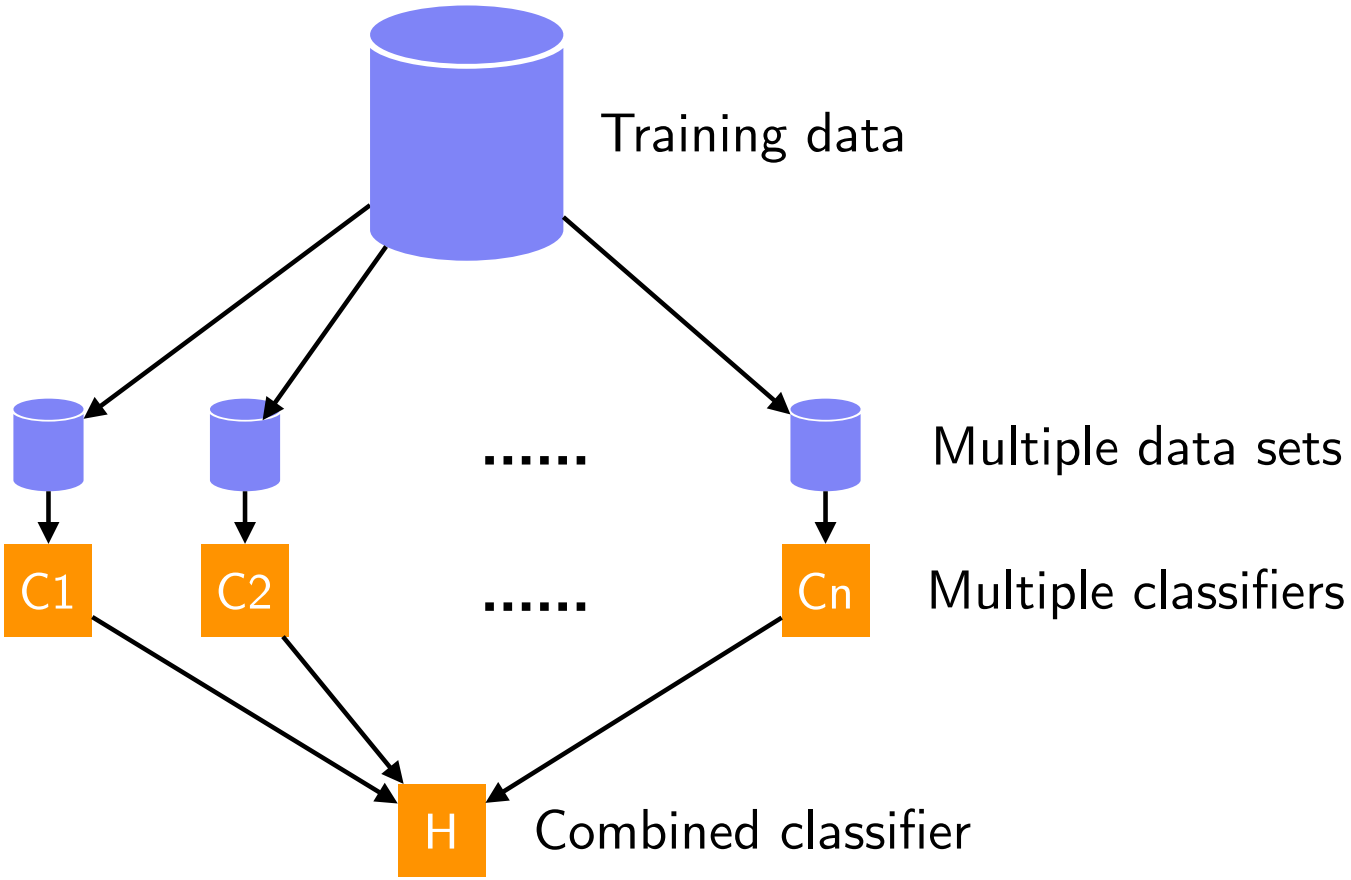
Reasons for using Ensembles

- Divide and conquer:
  - The given task may be too complex, or lie outside the space of functions that can be implemented by the chosen classifier method
  - Appropriate combinations of simple (e.g., linear) classifiers can learn complex (e.g., non-linear) boundaries

Reasons for using Ensembles

- Data fusion:
  - Several sets of data obtained from different sources, where the nature of features is different (e.g.: categorical and numerical features)
  - Data from each source can be used to train a different classifier, thus creating an ensemble boundaries

General idea



Components of an Ensemble

- A method to generate the individual classifiers of the ensemble
- A method for combining the outputs of these classifiers

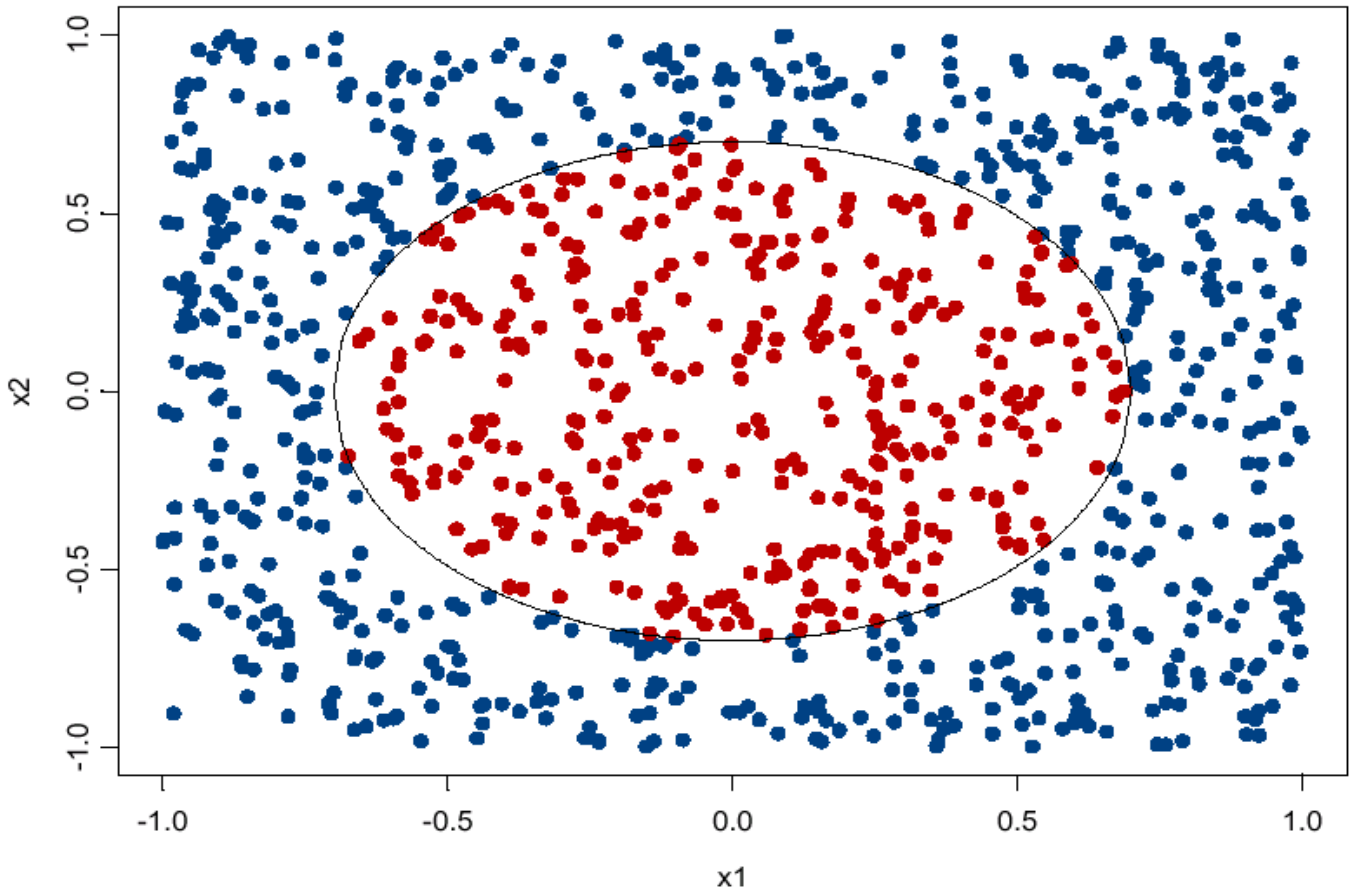
Diversity and Accuracy

- The individual classifiers must be diverse (errors on different data)
- If they make the same errors, such mistakes will be carried into the final prediction
- The component classifiers need to be "reasonably accurate" to avoid poor classifiers to obtain the majority of votes.

Bagging: Bootstrap Aggregation

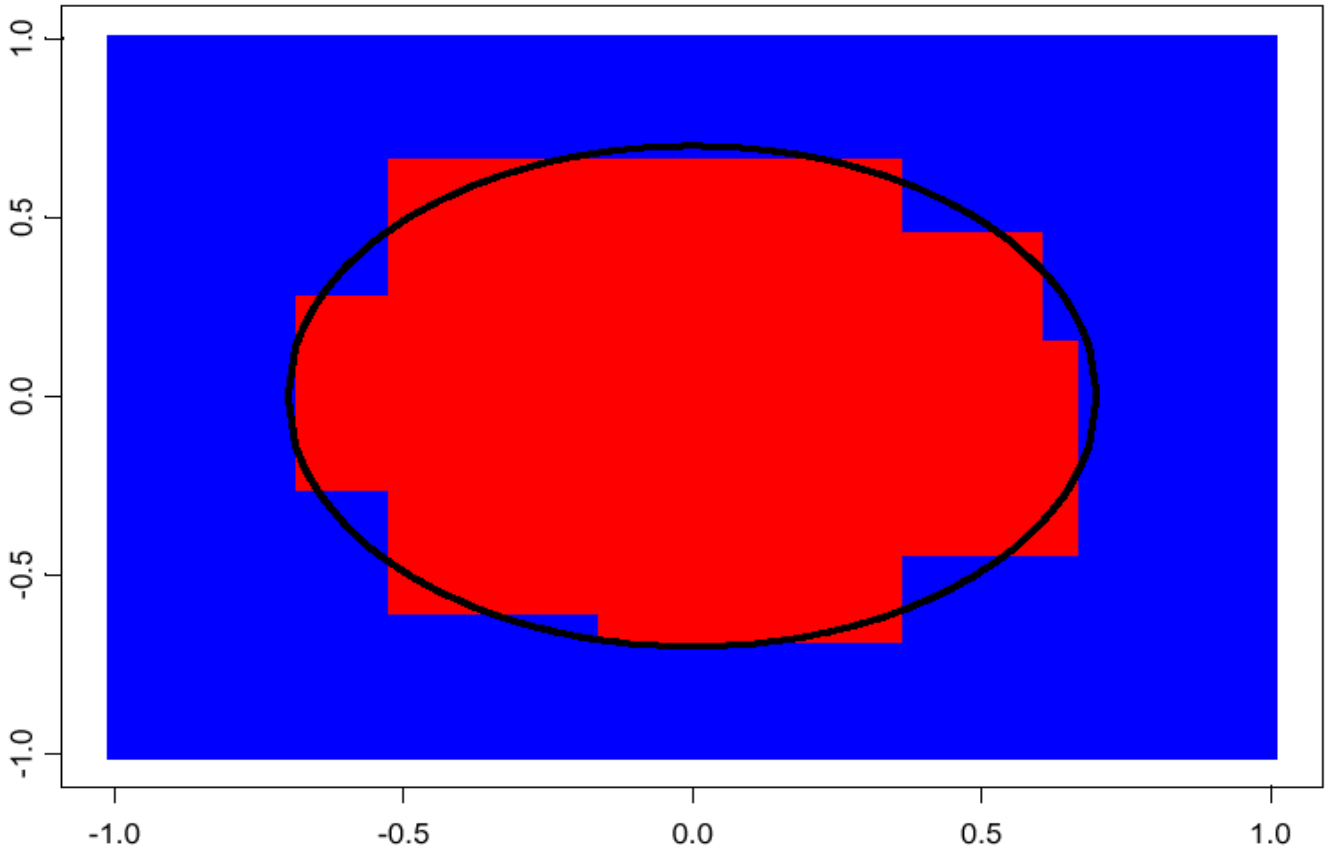
- Take repeated bootstrap samples from training set D (Breiman, 1994)
- Bootstrap sampling: Bootstrap sampling: Given set D containing N training examples, create D' by drawing N examples at random with replacement from D
- Bagging:
  - create k bootstrap samples $D_1, ..., D_k$
  - Train distinct classifier on each D_i
  - Classify new instances by majority vote/average

Bagging example

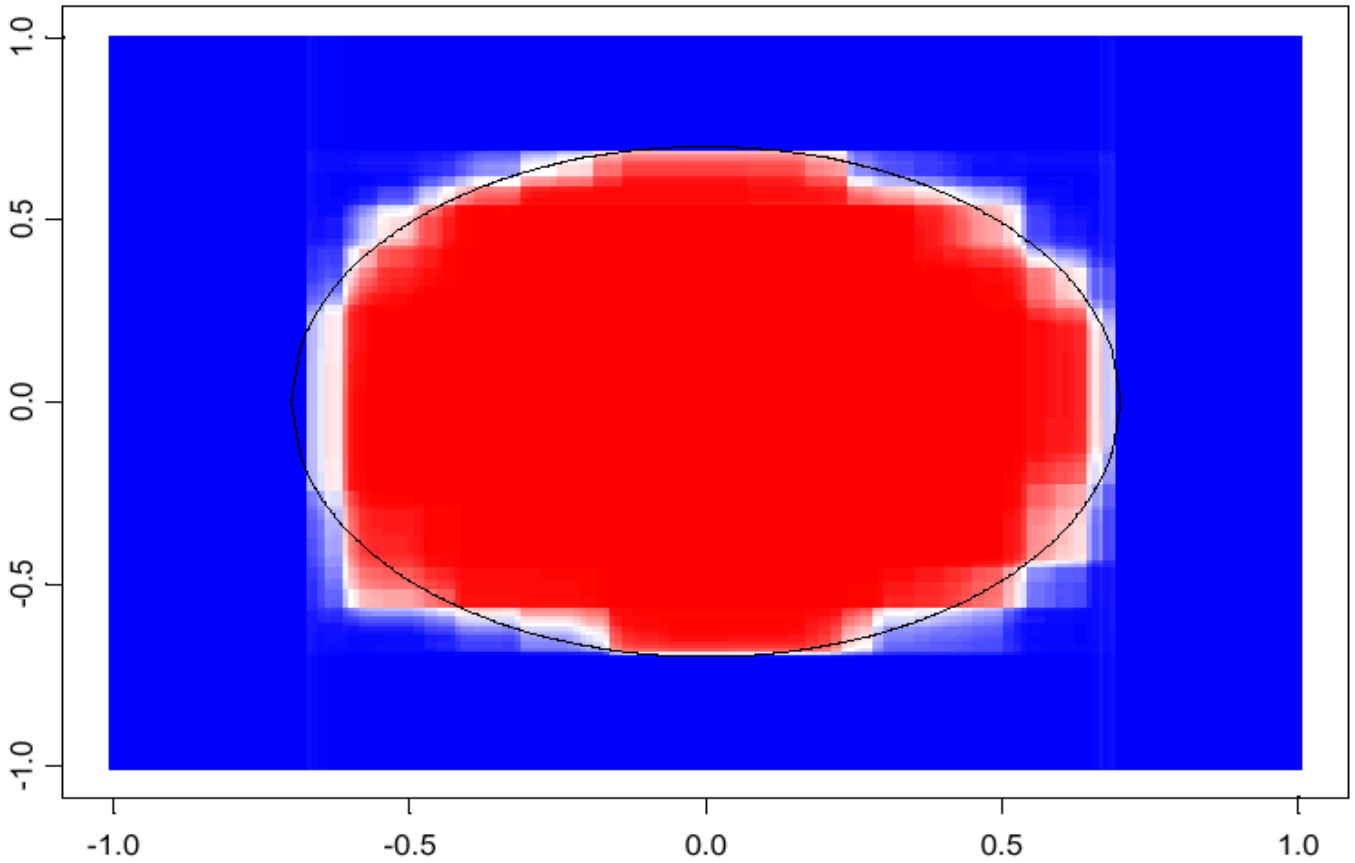


CART decision boundary

Decision tree learning algorithm; very similar to ID3



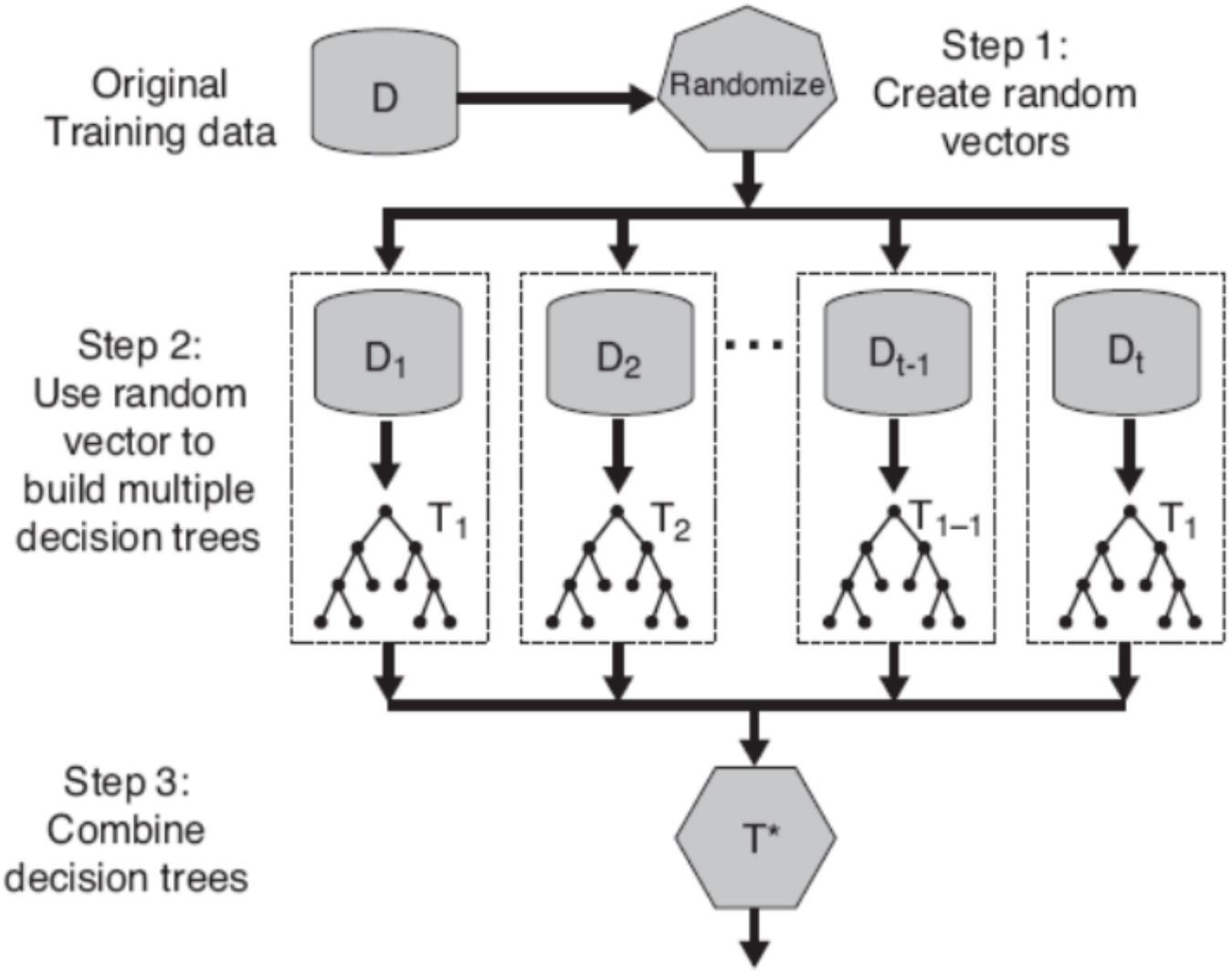
100 bagged trees



When is Bagging (bootstrapping) effective?

- To ensure diverse classifiers, the base classifier should be unstable, that is, small changes in the training set should lead to large changes in the classifier output.
- Large error reductions have been observed with decision trees and bagging. This is because decision trees are highly sensitive to small perturbations of the training data.
- Bagging is not effective with nearest neighbor classifiers. NN classifiers are highly stable with respect to variations of the training data.
- When the errors are highly correlated, and bagging becomes ineffective.

Random Forests



Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Two sources of randomness: "bagging" and "random input vectors"
- Use bootstrap aggregation to train many decision trees
  - Randomly subsample n examples
  - Train decision tree on subsample
  - Use average or majority vote among learned trees as prediction
- Also randomly subsample features: best split at each node is chosen from a random sample of m attributes instead of all attributes

Random forest algorithm

- For $b = 1$ to B
  - Draw a bootstrap sample of size N from the data
  - Grow a tree T_b using the bootstrap sample as follows
    - Choose m attributes uniformly at random from the data
    - Choose the best attribute among the m to split on
    - Split on the best attribute and recurse until partitions have fewer than s_{min} number of nodes
- Prediction for a new data point x
  - Regression: $\frac{1}{B}\sum_b T_b(x)$
  - Classification: choose the majority class label among $T_1(x), ..., T_B(x)$

Acknowledgements

Slides adapted from Dr. David Sontag's *Machine Learning* at MIT.