# DS 5220

## Supervised Machine Learning and Learning Theory

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 16 2019

# Logistics

- HW 3 is on Piazza, due on Oct. 25
- Exams
  - Midterm: Monday, Oct. 28
  - Final exam: Wednesday, Dec. 4
- Project
  - Proposal due on Oct. 21; teams of 2-3
  - Project presentation on Dec. 9
  - Project report due on Dec. 10
  - Project ideas and datasets posted on Piazza
  - Example projects from DS 4400 posted on Piazza

# Project Proposal

- Project Title
- Project Team
- Problem Description
  - What is the prediction problem you are trying to solve?
- Dataset
  - Link to data, brief description, number of records, feature dimensionality (at least 10K records)
- Approach and methodology
  - Normalization
  - Feature selection
  - Machine learning models you will try (at least 3)
  - Splitting into training and testing, cross validation
  - Language and packages you plan to use
- Metrics (how you will evaluate your models)

# Review: Naïve Bayes Classifier

- For each class label $k$
  1. Estimate prior $P[Y = k]$ from the data
  2. For each value $v$ of attribute $X_j$
     - Estimate $\mathrm{P}[X_j = v | Y = k]$

- Classify a new point via:

$$h(\mathbf{x}) = \arg\max_{y_k} \log P(Y = k) + \sum_{j=1}^{d} \log P(X_j = x_j \mid Y = k)$$

- In practice, the independence assumption doesn't often hold true, but Naïve Bayes performs very well despite it

# Review Naïve Bayes

- Density Estimators can estimate joint probability distribution from data
- Risk of overfitting and curse of dimensionality
- Naïve Bayes assumes that features are independent given labels
  - Reduces the complexity of density estimation
  - Even though the assumption is not always true, Naïve Bayes works well in practice
- Applications: text classification with bag-of-words representation
  - Naïve Bayes becomes a linear classifier

Generative model

# Outline

- Feature selection
  - Wrapper
  - Filter
  - Embedded methods
- Decision trees
  - Information Gain
  - ID3 algorithm
  - Pruning decision trees

# Supervised Learning Process

## Training

Data → Pre-processing → Feature extraction → Learning model

- Data: Labeled (Typically)
- Pre-processing: Normalization, Standardization
- Feature extraction: Feature Selection
- Learning model: Classification, Regression

## Testing

New data → Learning model → Predictions

- New data: Unlabeled
- Predictions:
  - Malicious / Benign — Classification
  - Risk score — Regression

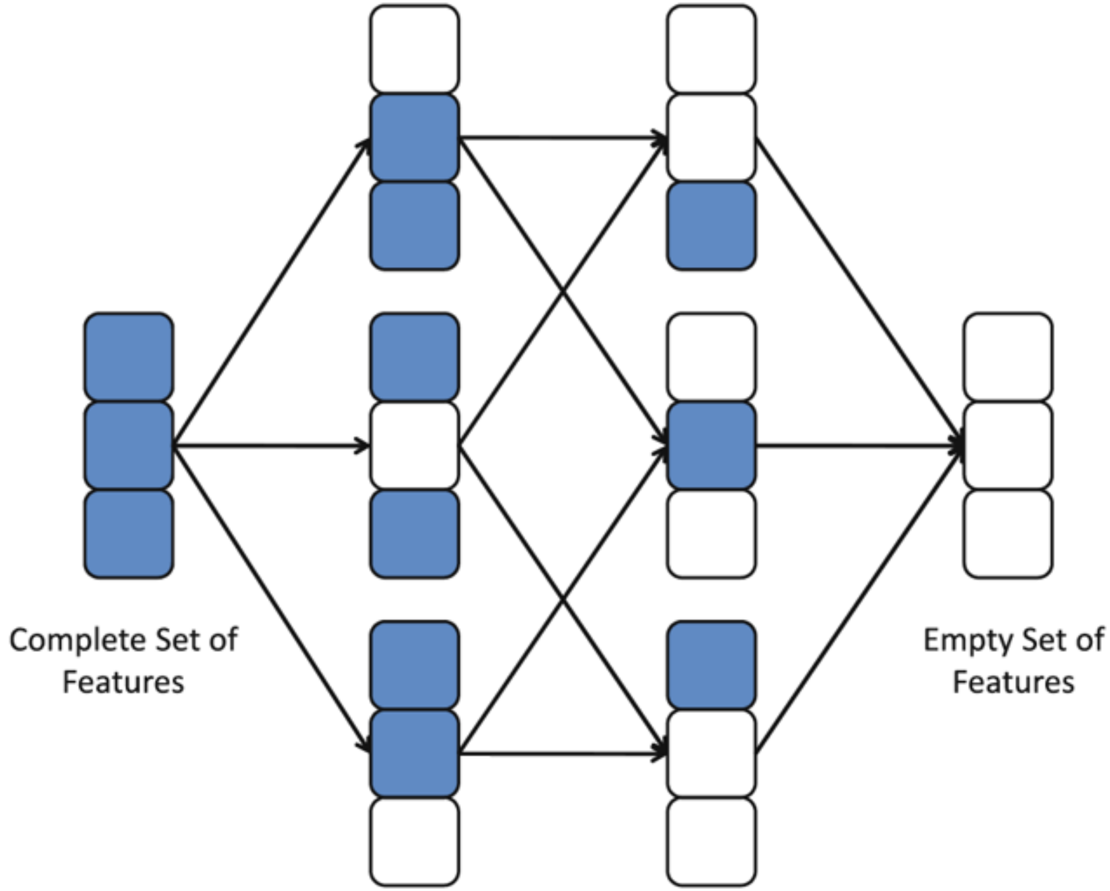
# Feature selection

- *Feature Selection*
  - Process for choosing an optimal subset of features according to a certain criteria

- Why we need Feature Selection:
  1. To improve performance (in terms of speed, predictive power, simplicity of the model).
  2. To visualize the data for model selection.
  3. To reduce dimensionality and remove noise.

# Feature Search Space

Search Space:




Exponentially large!

# Methods for Feature Selection

- Wrappers
  - Select subset of features that gives best prediction accuracy (using cross-validation)
  - Model-specific
- Filters
  - Compute some statistical metrics (correlation coefficient, mutual information)
  - Select features with statistics higher than threshold
- Embedded methods
  - Feature selection done as part of training
  - Example: Regularization (Lasso, L1 regularization)

# Feature Engineering

- Feature engineering is crucial to getting good results
- Strategy: overshoot and regularize
  - Define as many features as you can
  - Use regularization for models that support it
  - Use other feature selection methods (e.g., filters) otherwise
- Do cross-validation to evaluate selected features on multiple runs
- When feature selection is frozen, evaluate on test set

# Wrappers: Search Strategy

- With an **exhaustive search**

$$1011100000010001000010000000000100101010$$

With $d$ features $\rightarrow$ $2^d$ possible feature subsets.

20 features … 1 million feature sets to check
25 features … 33.5 million sets
30 features … 1.1 billion sets

- Need for **a search strategy**
  - Sequential forward selection
  - Recursive backward elimination
  - Genetic algorithms
  - Simulated annealing
  - …

# Wrappers: Sequential Forward Selection

**Start** with the empty set $S = \emptyset$

**While** *stopping criteria not met*

**For** each feature $X_f$ not in $S$

- Define $S' = S \cup \{X_f\}$
- Train model using the features in $S'$
- Compute the accuracy on validation set

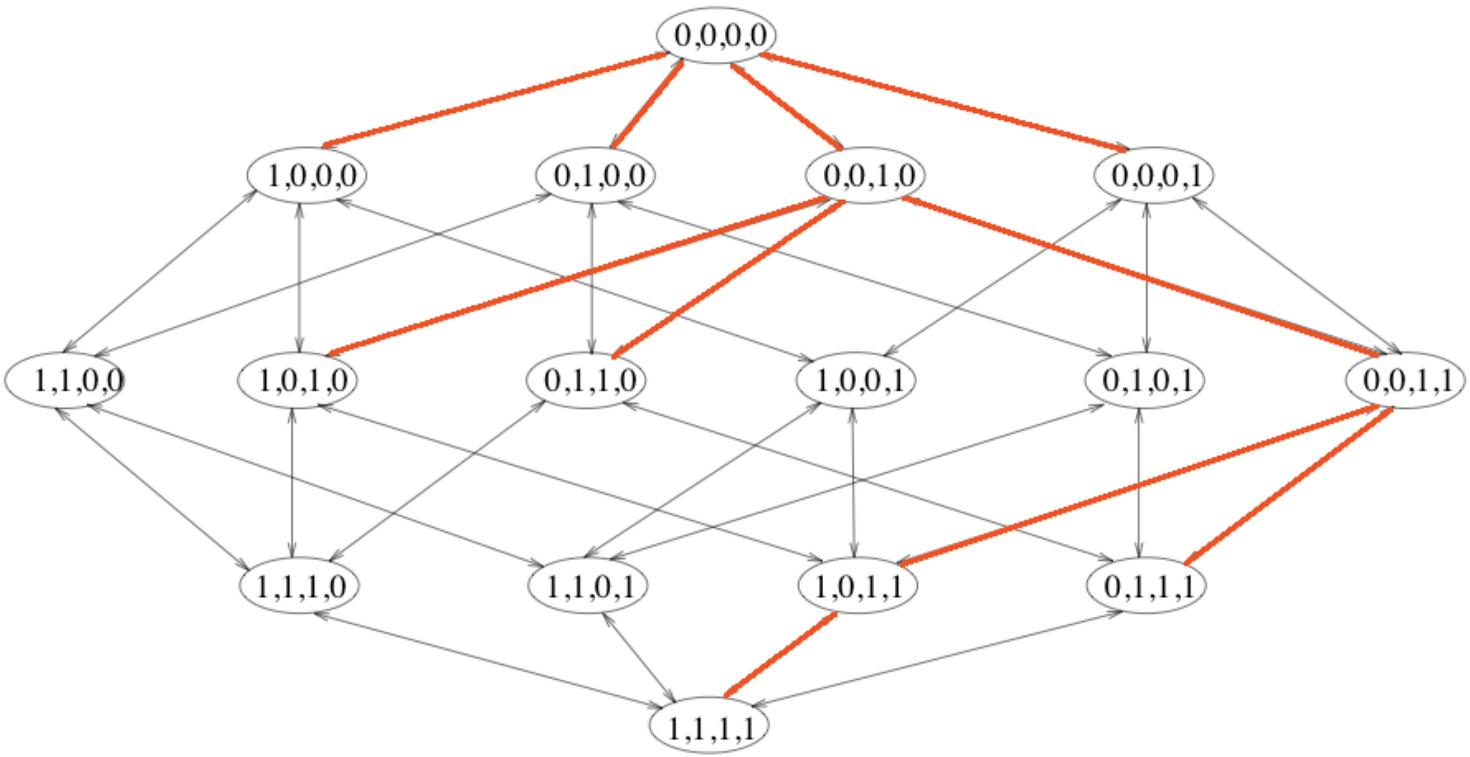
**End**

$S = S'$ where $S'$ is the feature set with the greatest accuracy

**End**

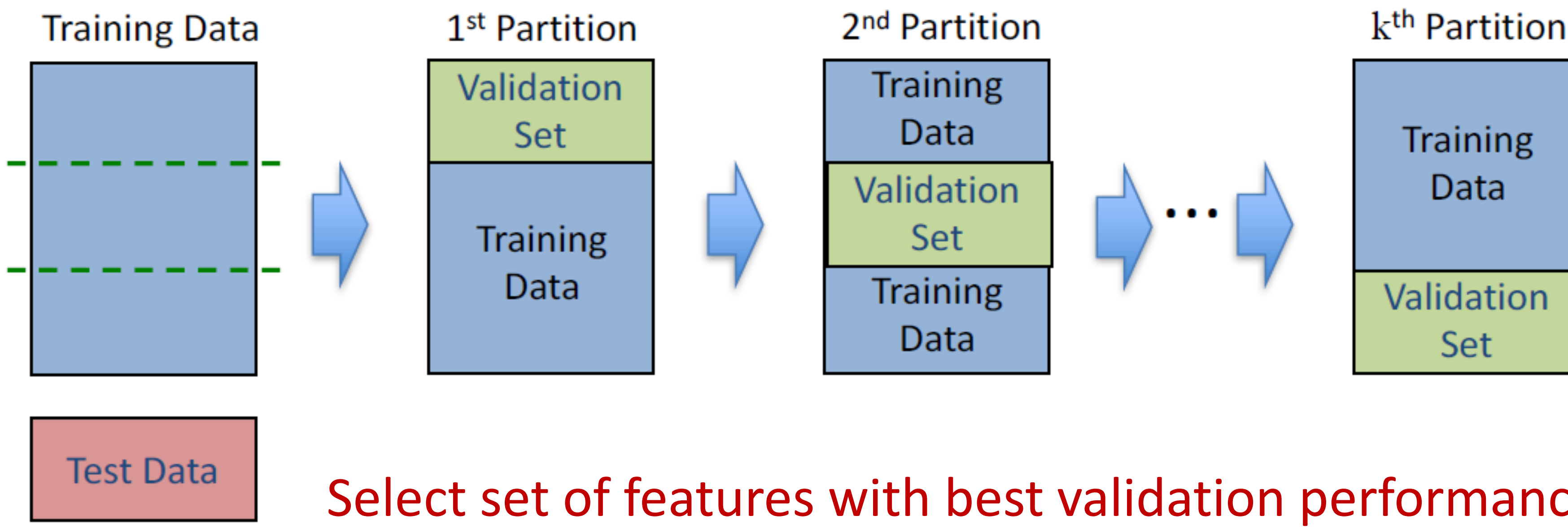
Backward feature selection starts with all features and eliminates backward

# Search complexity for sequential forward selection




- Evaluates $\frac{d(d+1)}{2}$ features sets instead of $2^d$

# Cross Validation




Select set of features with best validation performance

- k-fold CV
  - Split data into k partitions of equal size
- Leave-one-out CV (LOOCV)
  - k=n (validation set only one point)

# Filters

**Principle**: *replace evaluation of model with quick to compute statistics* $J(X_f)$

| $k$ | $J(X_k)$ |
|---|---|
| 35 | 0.846 |
| 42 | 0.811 |
| 10 | 0.810 |
| 654 | 0.611 |
| 22 | 0.443 |
| 59 | 0.388 |
| ... | ... |
| 212 | 0.09 |
| 39 | 0.05 |

**For** each feature $X_f$

- Compute $J(X_f)$

**End**

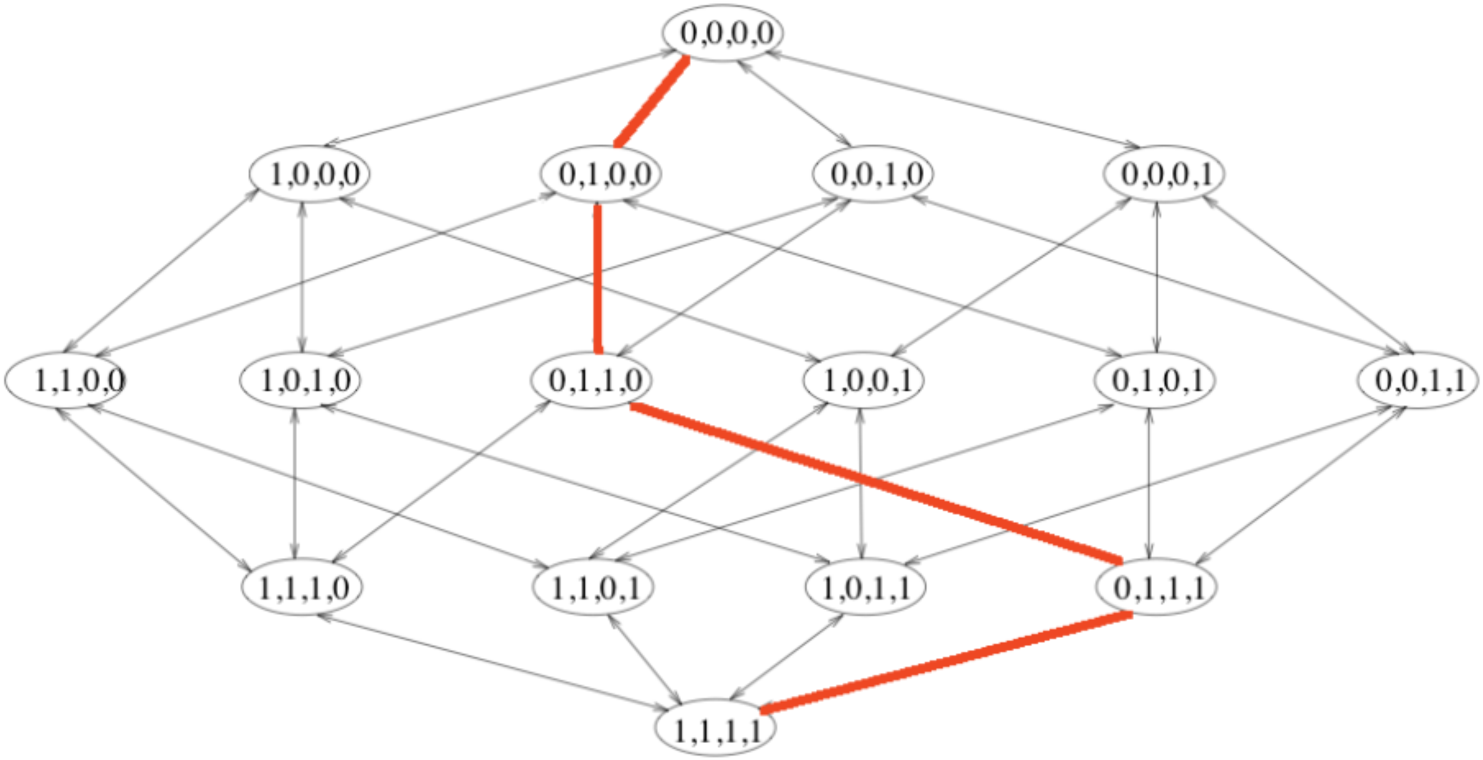
Rank features according to $J(X_f)$

Choose manual cut-off point

**Examples of filtering criterion**

- The mutual information with the target variable $J(X_f) = I(X_f; Y)$
- The correlation with the target variable
- $\chi^2$ - statistic

# Search Complexity for Filter Methods




**Pros**:
- A lot less expensive!

**Cons**:
- Not model-oriented

# Embedded methods: Regularization

## Lasso regression

$$J(\theta) = \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{Squared Residuals}} + \lambda \underbrace{\sum_{j=1}^{d} |\theta_j|}_{\text{Regularization}}$$

- L1 norm for regularization
- No closed form solution
- Algorithms based on sub-gradient descent

# Embedded methods: Regularization

**Principle**: the classifier performs feature selection as part of the learning procedure

**Example**: the logistic LASSO (Tibshirani, 1996)

$$f(\boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x})}} = P(Y = 1|\boldsymbol{x})$$

With Error Function:

$$E = \underbrace{-\sum_{i=1}^{N} \{y_i \log f(\boldsymbol{x}_i) + (1 - y_i) \log(1 - f(\boldsymbol{x}_i))\}}_{\text{Cross-entropy error}} \underbrace{+ \lambda \sum_{f=1}^{d} |w_f|}_{\text{Regularizing term}}$$

**Pros**:
- Performs feature selection as part of learning the procedure

**Cons**:
- Computationally demanding

# Summary: Feature Selection


- Filtering
- $L_1$ regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

# Summary: Feature Selection


- *Filtering*
- $L_1$ regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Good preprocessing step
- Fails to capture relationship between features

# Summary: Feature Selection


- Filtering
- *L₁ regularization (embedded methods)*
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Can add regularization in optimization objective
- Can be solved with Gradient Descent
- Can be applied to many models (e.g., linear or logistic regression)
- Can not be applied to all methods (e.g., kNN)

# Summary: Feature Selection


- Filtering
- $L_1$ regularization (embedded methods)
- *Wrappers*
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Most directly optimize prediction performance
- Can be very expensive, even with greedy search methods
- Cross-validation is a good objective function to start with

# Outline

- Feature selection
  - Wrapper
  - Filter
  - Embedded methods
- Decision trees
  - Information Gain
  - ID3 algorithm
  - Pruning decision trees
- Lab decision trees

# Sample Dataset

- Columns denote features $X_i$
- Rows denote labeled instances $< x_i, y_i >$
- Class label denotes whether a tennis game was played

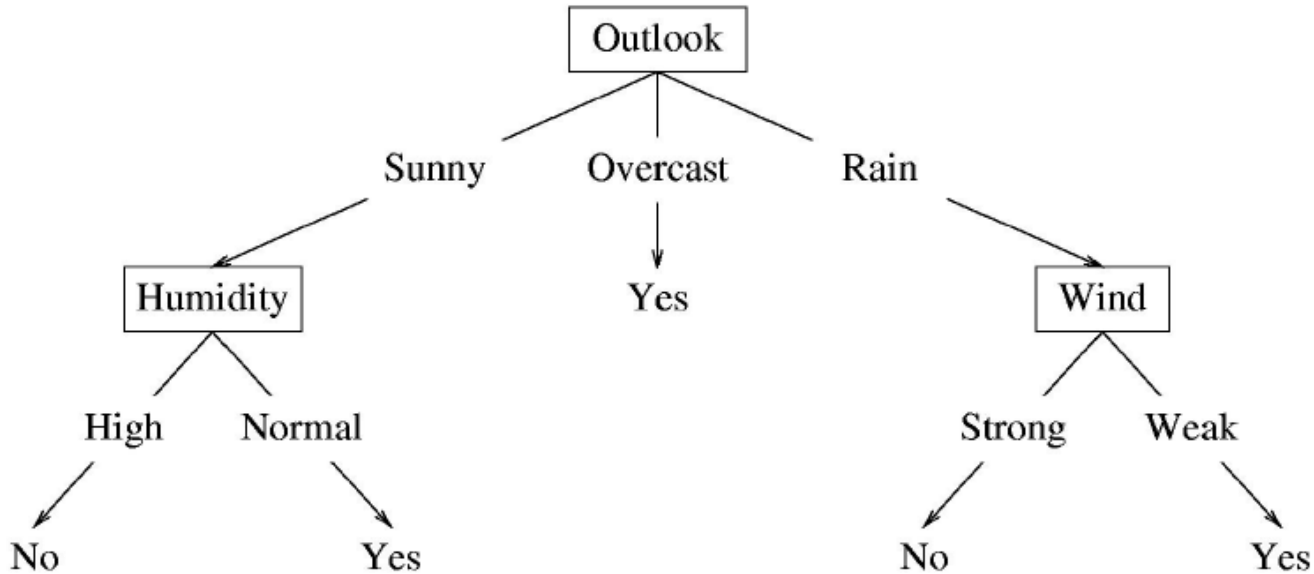
$< x_i, y_i >$

Categorical data

| Predictors | | | | Response |
|---|---|---|---|---|
| **Outlook** | **Temperature** | **Humidity** | **Wind** | **Class** |
| Sunny | Hot | High | Weak | No |
| Sunny | Hot | High | Strong | No |
| Overcast | Hot | High | Weak | Yes |
| Rain | Mild | High | Weak | Yes |
| Rain | Cool | Normal | Weak | Yes |
| Rain | Cool | Normal | Strong | No |
| Overcast | Cool | Normal | Strong | Yes |
| Sunny | Mild | High | Weak | No |
| Sunny | Cool | Normal | Weak | Yes |
| Rain | Mild | Normal | Weak | Yes |
| Sunny | Mild | Normal | Strong | Yes |
| Overcast | Mild | High | Strong | Yes |
| Overcast | Hot | Normal | Weak | Yes |
| Rain | Mild | High | Strong | No |

# Decision Tree

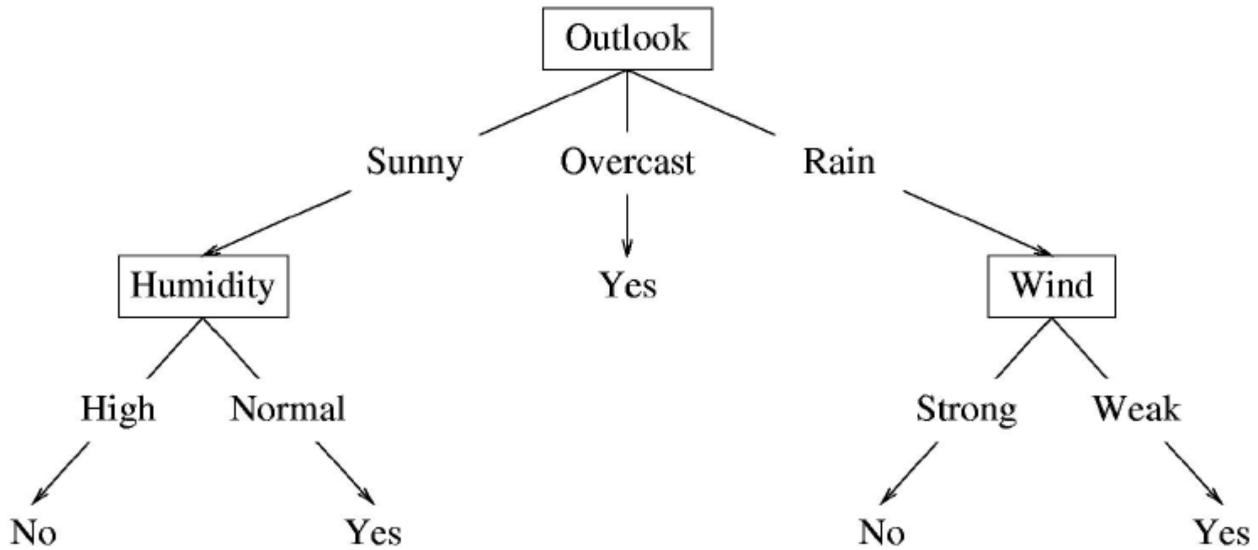
- A possible decision tree for the data:




- Each internal node: test one attribute $X_i$
- Each branch from a node: selects one value for $X_i$
- Each leaf node: predict $Y$ (or $p(Y \mid x \in \mathrm{leaf})$ )

# Decision Tree

- A possible decision tree for the data:

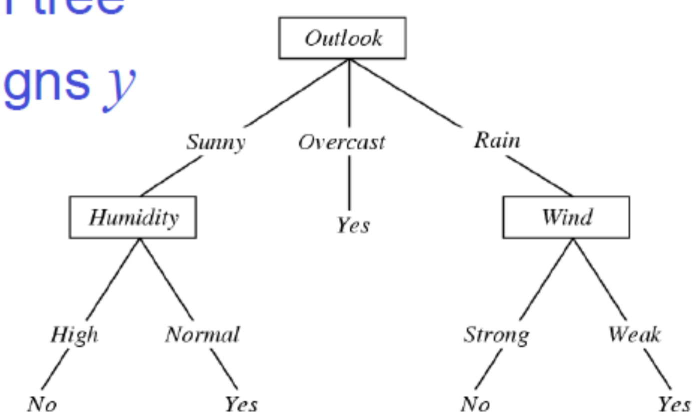



- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

# Decision Tree Learning

**Problem Setting**:

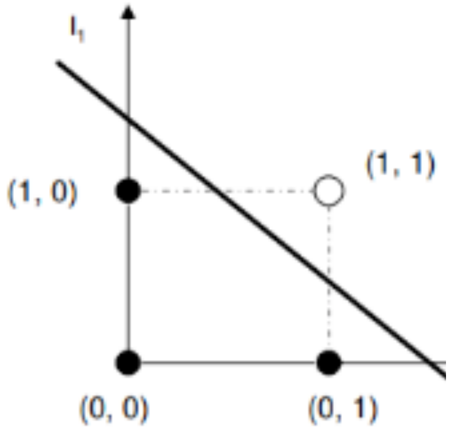
- Set of possible instances $X$
  - each instance $x$ in $X$ is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - *Y* is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis $h$ is a decision tree
  - trees sorts $x$ to leaf, which assigns $y$



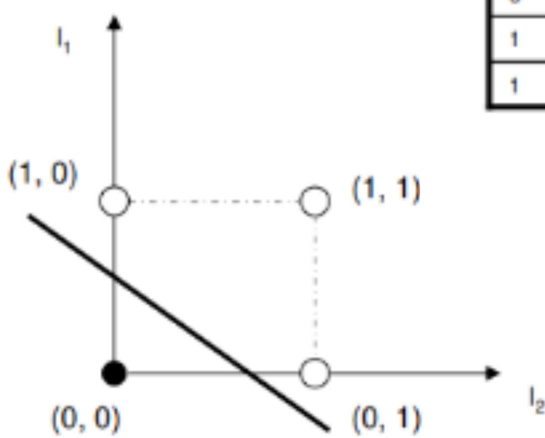

Slide by Tom Mitchell

# Learning Boolean Functions

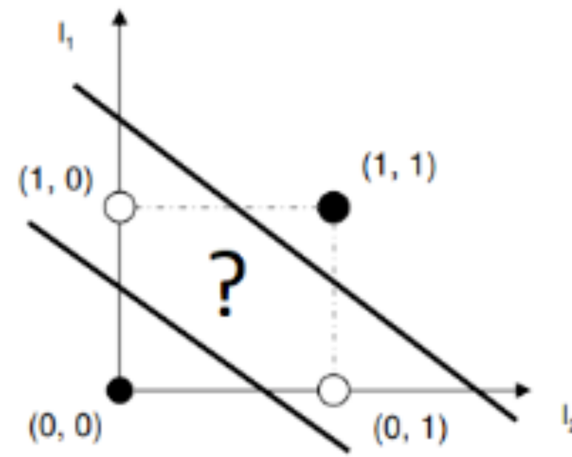
| AND | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




| OR | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




| XOR | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

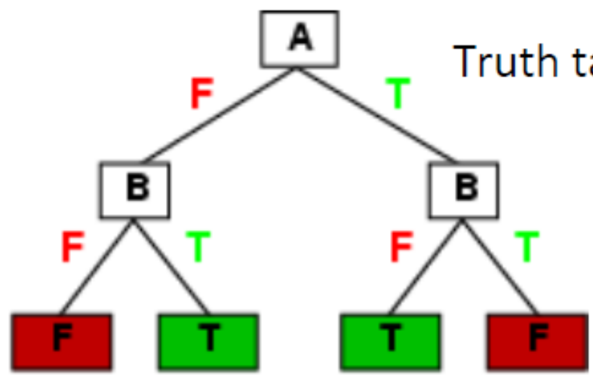



XOR cannot be learned with linear classifiers

# Expressiveness

- Decision trees can represent any boolean function of the input attributes

| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |




Truth table row → path to leaf

- In the worst case, the tree will require exponentially many nodes

# Occam's Razor

- Principle stated by William of Ockham (1285-1347)
  - "*non sunt multiplicanda entia praeter necessitatem* "
  - entities are not to be multiplied beyond necessity
  - AKA Occam's Razor, Law of Economy, or Law of Parsimony

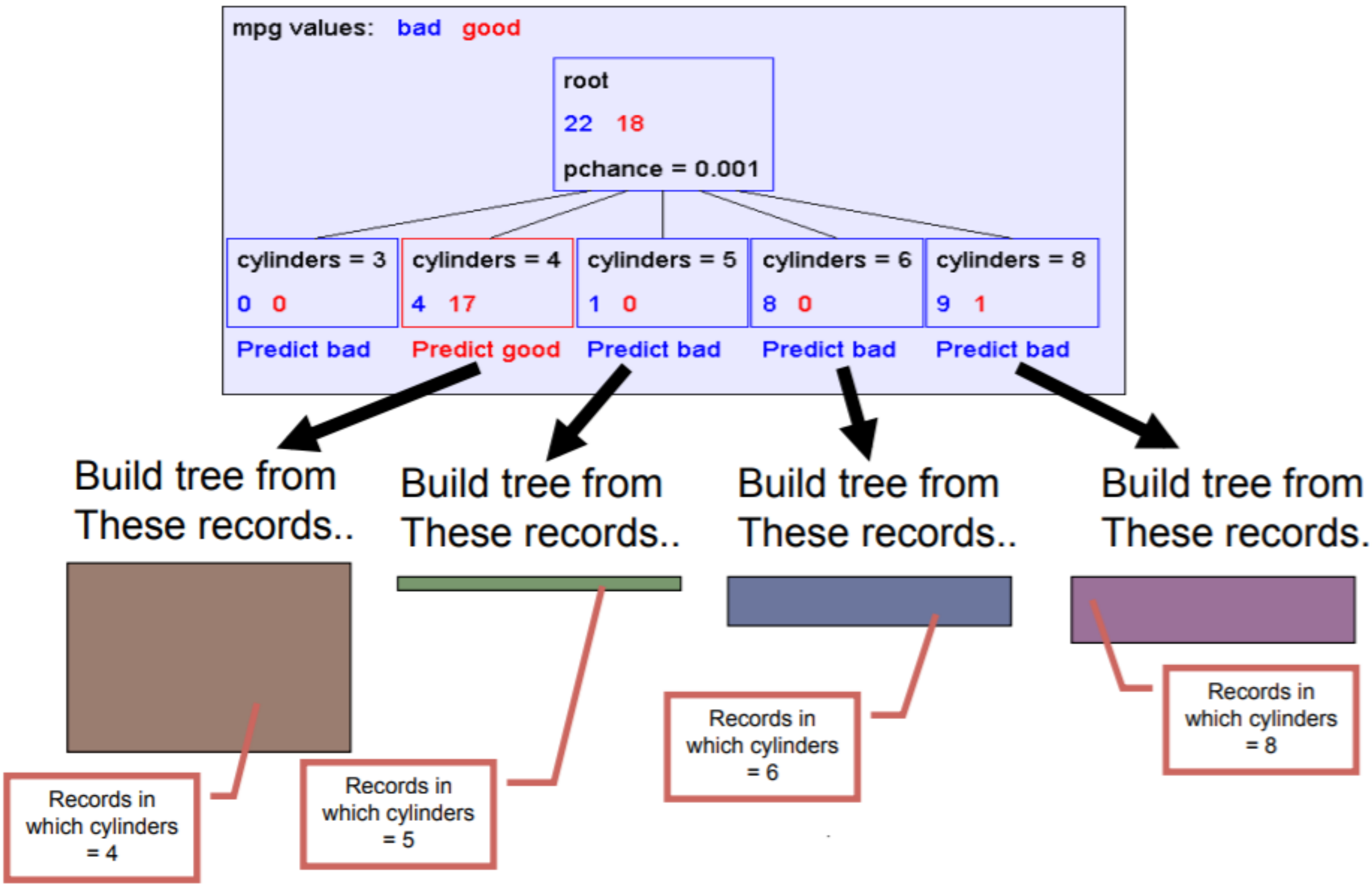
**Idea**: The simplest consistent explanation is the best

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

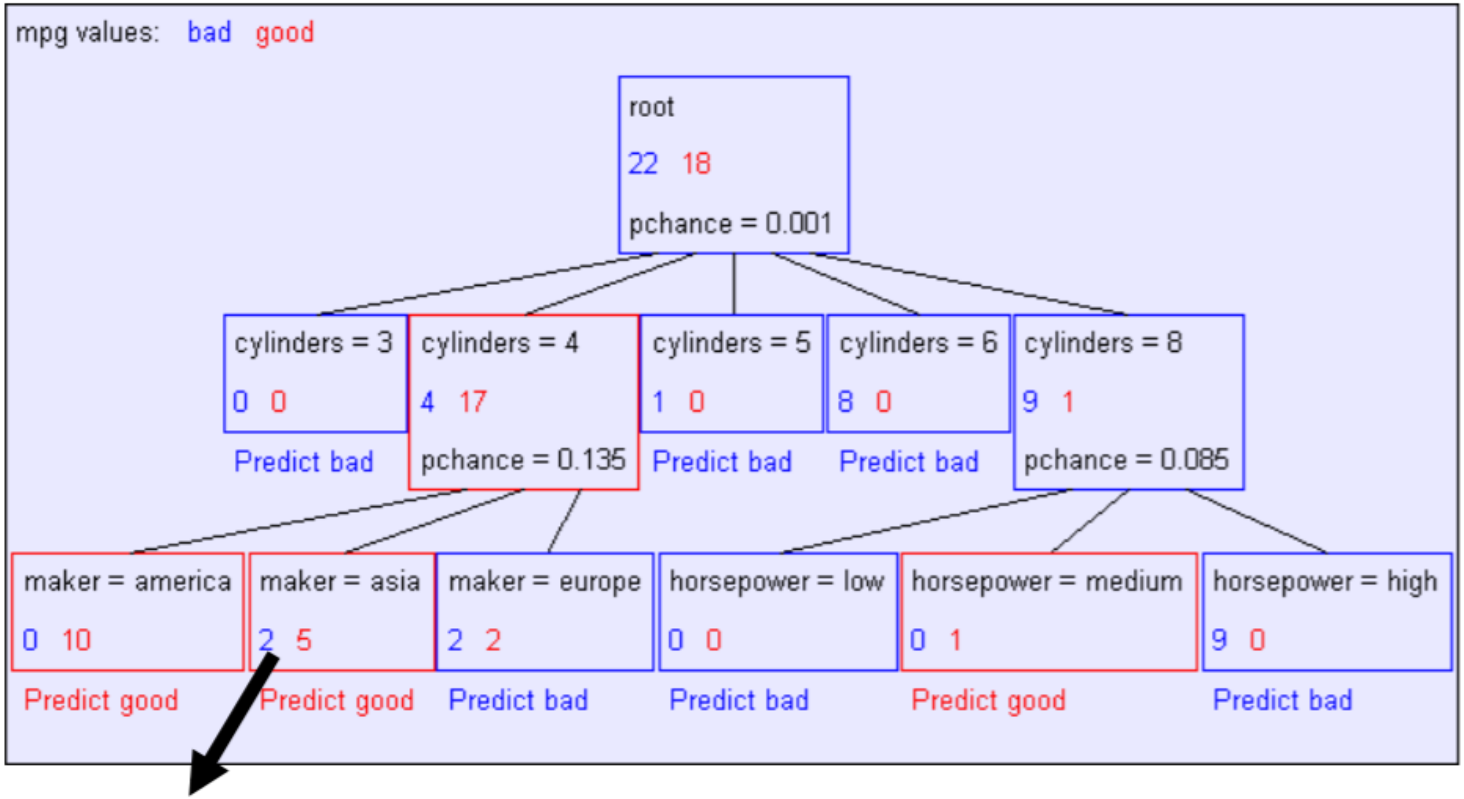
# Learning Decision Trees

- 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

# Key Idea: Use Recursion Greedily



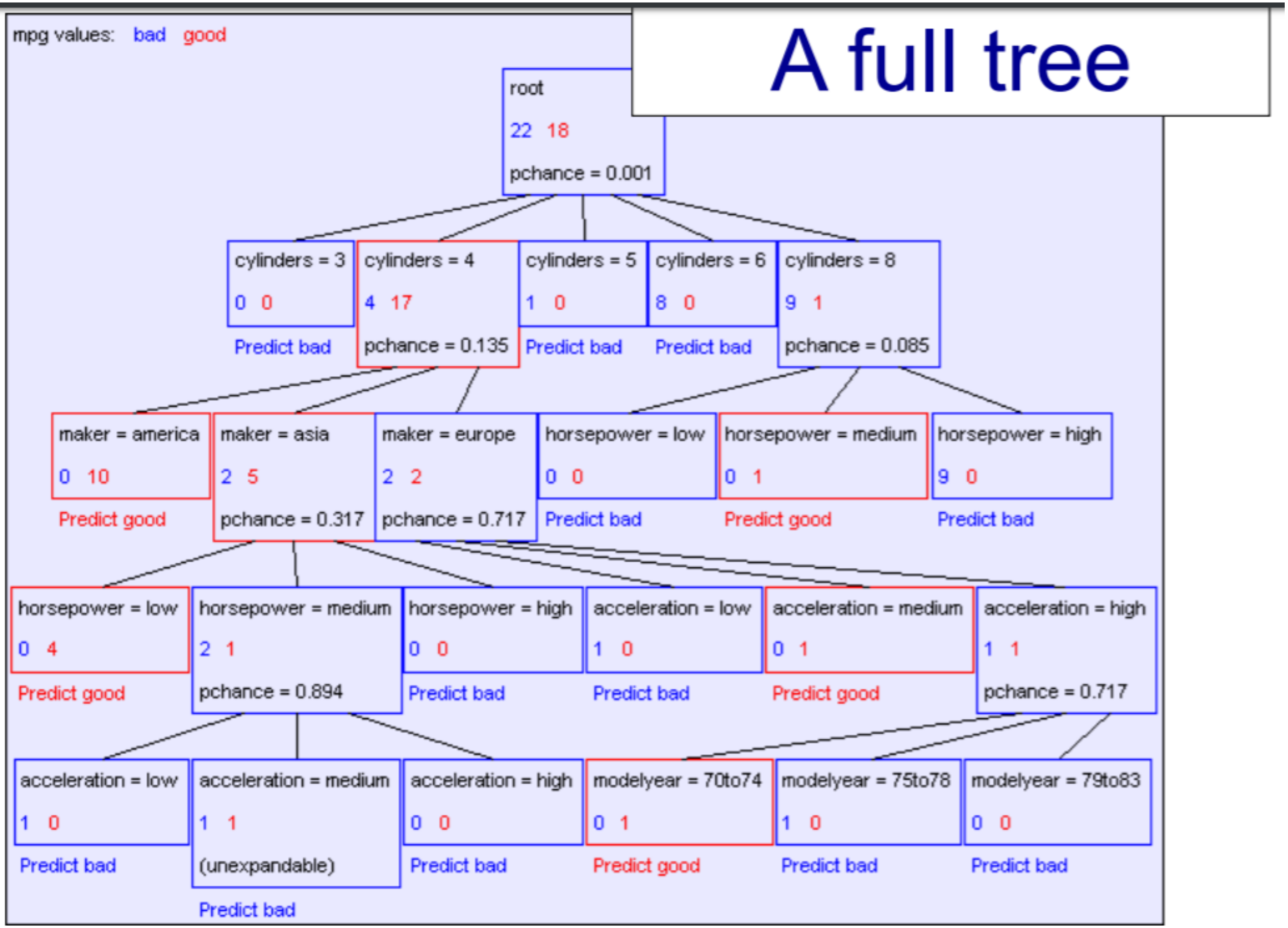
# Second Level




Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

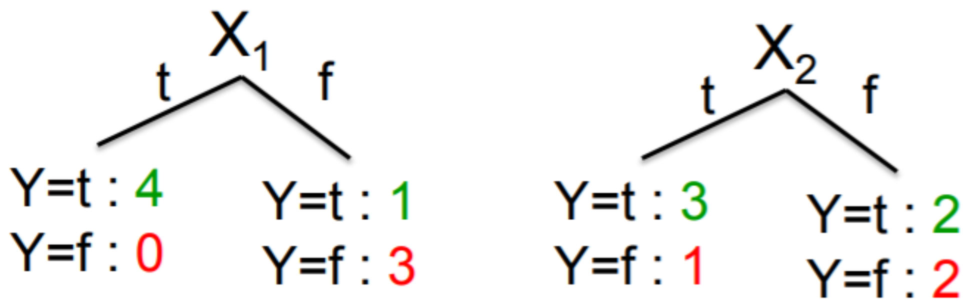
(Similar recursion in the other cases)

# Full Tree


mpg values: bad good
A full tree
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad

# Splitting

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 at leaves to define probability distributions, so we can measure uncertainty!

Use entropy-based measure (Information Gain)

# Transmitting Bits

You are watching a set of independent random samples of X

You see that X has four possible values

| | | | |
|---|---|---|---|
| P(X=A) = 1/4 | P(X=B) = 1/4 | P(X=C) = 1/4 | P(X=D) = 1/4 |

So you might see: BAACBADCDADDDA…

You transmit data over a binary serial link. You can encode each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

0100001001001110110011111100…

# Use Fewer Bits

Someone tells you that the probabilities are not equal

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

## It's possible…

…to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

# Use Fewer Bits

Someone tells you that the probabilities are not equal

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

It's possible…

…to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

| | |
|---|---|
| A | 0 |
| B | 10 |
| C | 110 |
| D | 111 |

(This is just one of several ways)

# General case

Suppose X can have one of *m* values… $V_1, V_2, \ldots V_m$

| | | | |
|---|---|---|---|
| $P(X=V_1) = p_1$ | $P(X=V_2) = p_2$ | …. | $P(X=V_m) = p_m$ |

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution? It's

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$

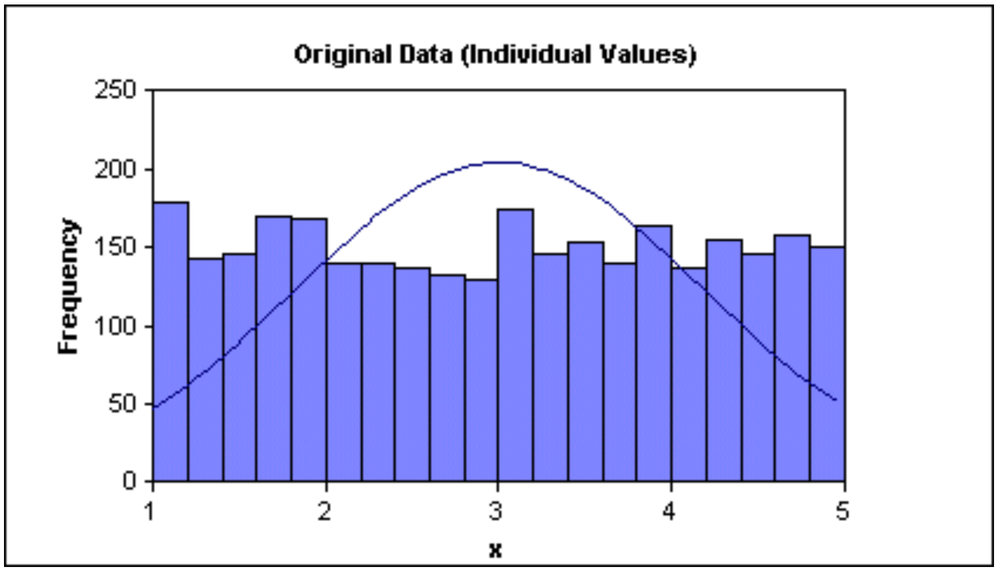
$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$

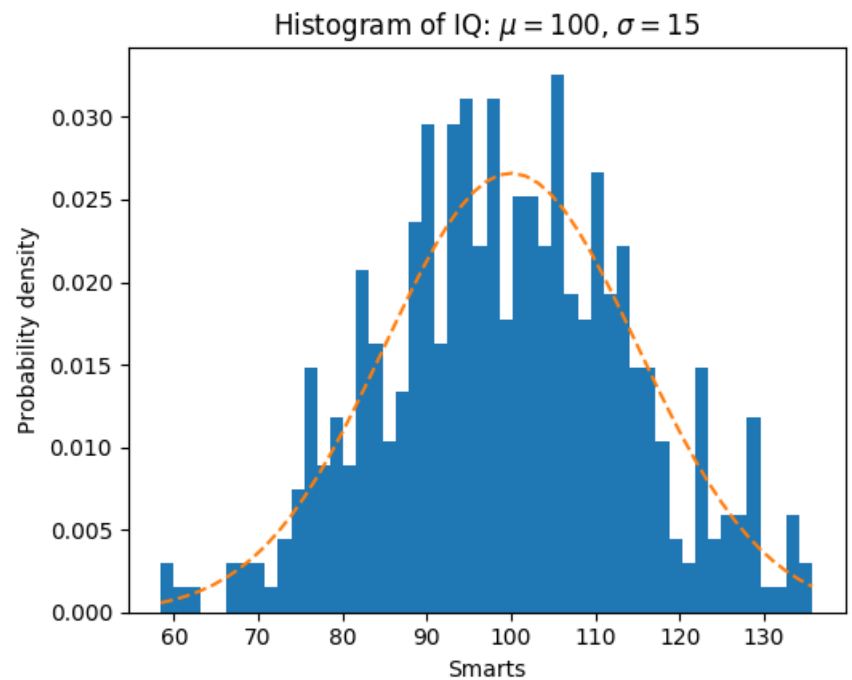
H(X) = The entropy of X

- "High Entropy" means X is from a uniform (boring) distribution
- "Low Entropy" means X is from varied (peaks and valleys) distribution

# High/Low Entropy

Which distribution has high entropy?




High

Low

# Conditional Entropy

**Suppose I'm trying to predict output Y and I have input X**

**X = College Major**

**Y = Likes "Gladiator"**

| X | Y |
|---|---|
| Math | Yes |
| History | No |
| CS | Yes |
| Math | No |
| Math | No |
| CS | Yes |
| History | No |
| Math | Yes |

**Let's assume this reflects the true probabilities**

**E.G. From this data we estimate**

- *P(LikeG = Yes) = 0.5*
- *P(Major = Math & LikeG = No) = 0.25*
- *P(Major = Math) = 0.5*
- *P(LikeG = Yes | Major = History) = 0*

**Note:**

- *H(X) = 1.5*
- *H(Y) = 1*

# Conditional Entropy

X = College Major

Y = Likes "Gladiator"

| X | Y |
|---|---|
| Math | Yes |
| History | No |
| CS | Yes |
| Math | No |
| Math | No |
| CS | Yes |
| History | No |
| Math | Yes |

**Definition of Specific Conditional Entropy:**

$H(Y|X=v)$ = **The entropy of** $Y$ **among only those records in which** $X$ **has value** $v$

## Example:

- $H(Y|X=Math) = 1$
- $H(Y|X=History) = 0$
- $H(Y|X=CS) = 0$

# Conditional Entropy

**X = College Major**

**Y = Likes "Gladiator"**

| X | Y |
|---|---|
| Math | Yes |
| History | No |
| CS | Yes |
| Math | No |
| Math | No |
| CS | Yes |
| History | No |
| Math | Yes |

**Definition of Conditional Entropy:**

$H(Y|X)$ = The average specific conditional entropy of $Y$

= if you choose a record at random what will be the conditional entropy of $Y$, conditioned on that row's value of $X$

= Expected number of bits to transmit $Y$ if both sides will know the value of $X$

$= \Sigma_j \, Prob(X=v_j) \, H(Y \mid X = v_j)$

# Conditional Entropy

**X = College Major**

**Y = Likes "Gladiator"**

| X | Y |
|---|---|
| Math | Yes |
| History | No |
| CS | Yes |
| Math | No |
| Math | No |
| CS | Yes |
| History | No |
| Math | Yes |

**Definition of Conditional Entropy:**

$H(Y|X)$ = The average conditional entropy of $Y$

$= \Sigma_j Prob(X=v_j)\ H(Y \mid X = v_j)$

**Example:**

| $v_j$ | $Prob(X=v_j)$ | $H(Y \mid X = v_j)$ |
|---|---|---|
| Math | 0.5 | 1 |
| History | 0.25 | 0 |
| CS | 0.25 | 0 |

$H(Y|X) = 0.5 * 1 + 0.25 * 0 + 0.25 * 0 = 0.5$

# Information Gain

**X = College Major**

**Y = Likes "Gladiator"**

| X | Y |
|---|---|
| Math | Yes |
| History | No |
| CS | Yes |
| Math | No |
| Math | No |
| CS | Yes |
| History | No |
| Math | Yes |

**Definition of Information Gain:**

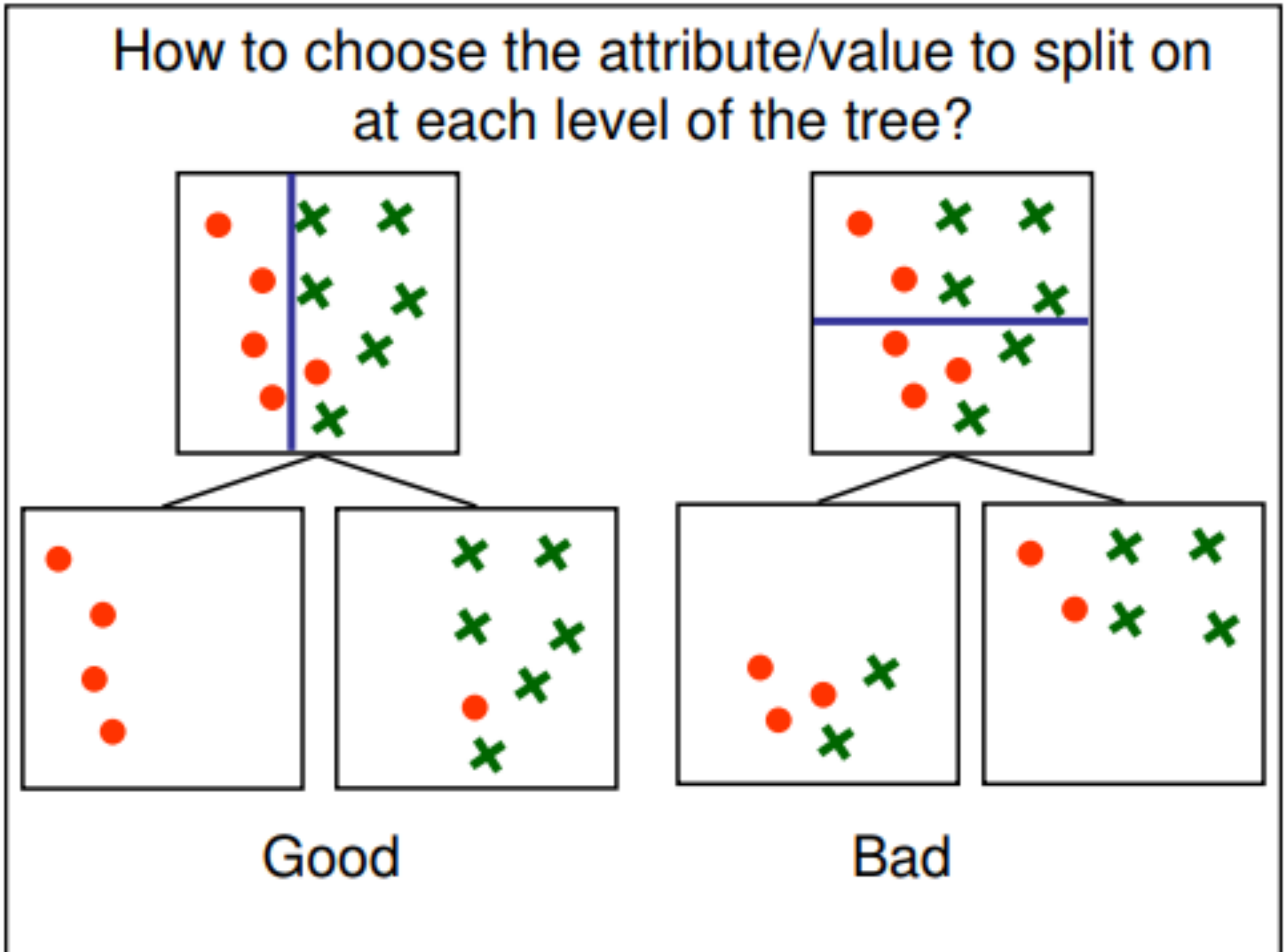
$IG(Y|X)$ = **I must transmit** $Y$. **How many bits on average would it save me if both ends of the line knew** $X$**?**

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

**Example:**

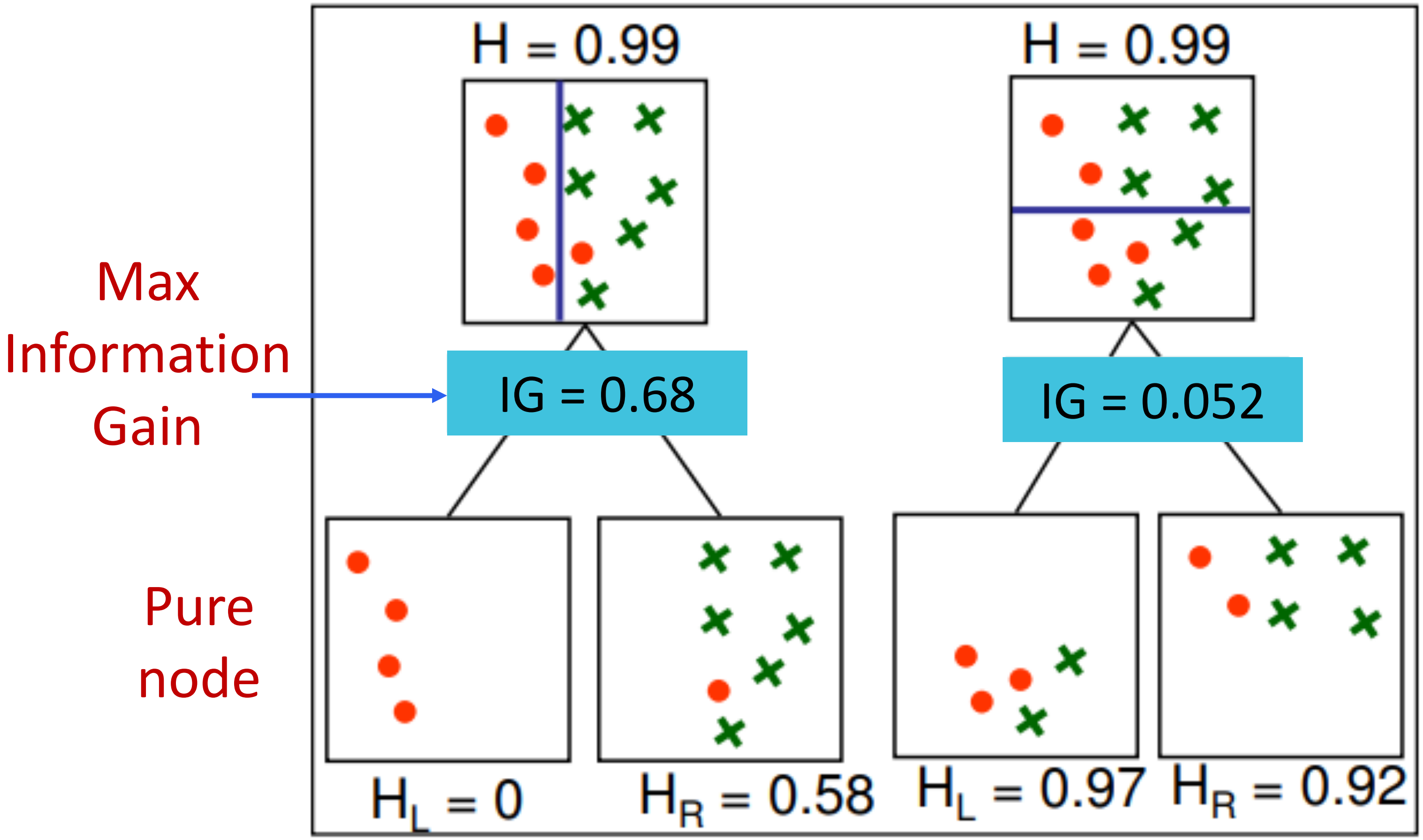
- **H(Y) = 1**
- **H(Y|X) = 0.5**
- **Thus IG(Y|X) = 1 – 0.5 = 0.5**

# Example



How to choose the attribute/value to split on at each level of the tree?
Good
Bad

# Example Information Gain



# Learning Decision Trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

- Recurse

ID3 algorithm uses Information Gain
Information Gain reduces uncertainty on Y

# Acknowledgements


- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!