Big Data Analysis: Classification - Decision Tree

Younghoon Kim

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

Decision Trees on MapReduce

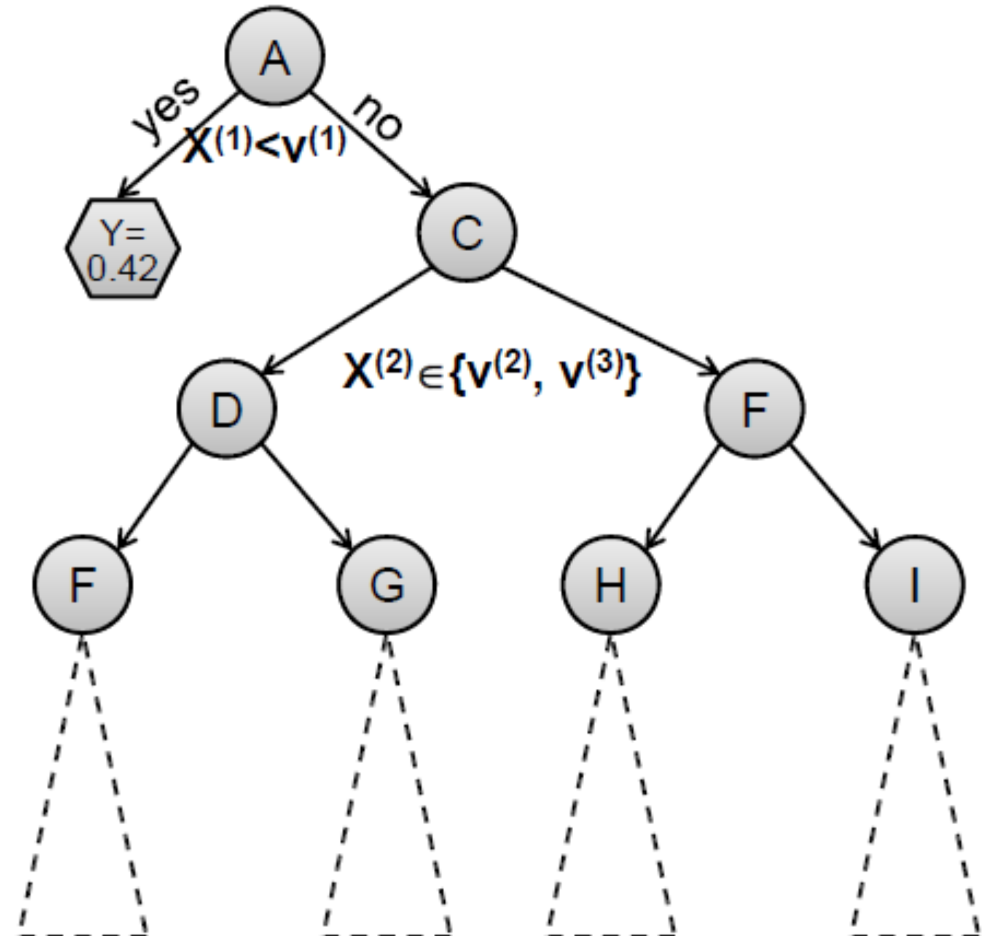
CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

Classification

- Given
  - A set of d-dimensional vectors: $x_i = \langle x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d)} \rangle$
  - O_j: domain of $x^{(j)}$
    - Categorical: e.g., O_j = {red, blue, orange, green}
    - Numerical: $O_j = \{1, 2, \ldots, 10\}$ or R
  - y_i: a label in a categorical domain O_Y
    - E.g., {yes, no} or { +, - }
  - Data D: $\{x_i, y_i\}_{i=1,\ldots,n}$
- Goal
  - Given a d-dimensional vector x, whose y is unknown,
  - Predict y

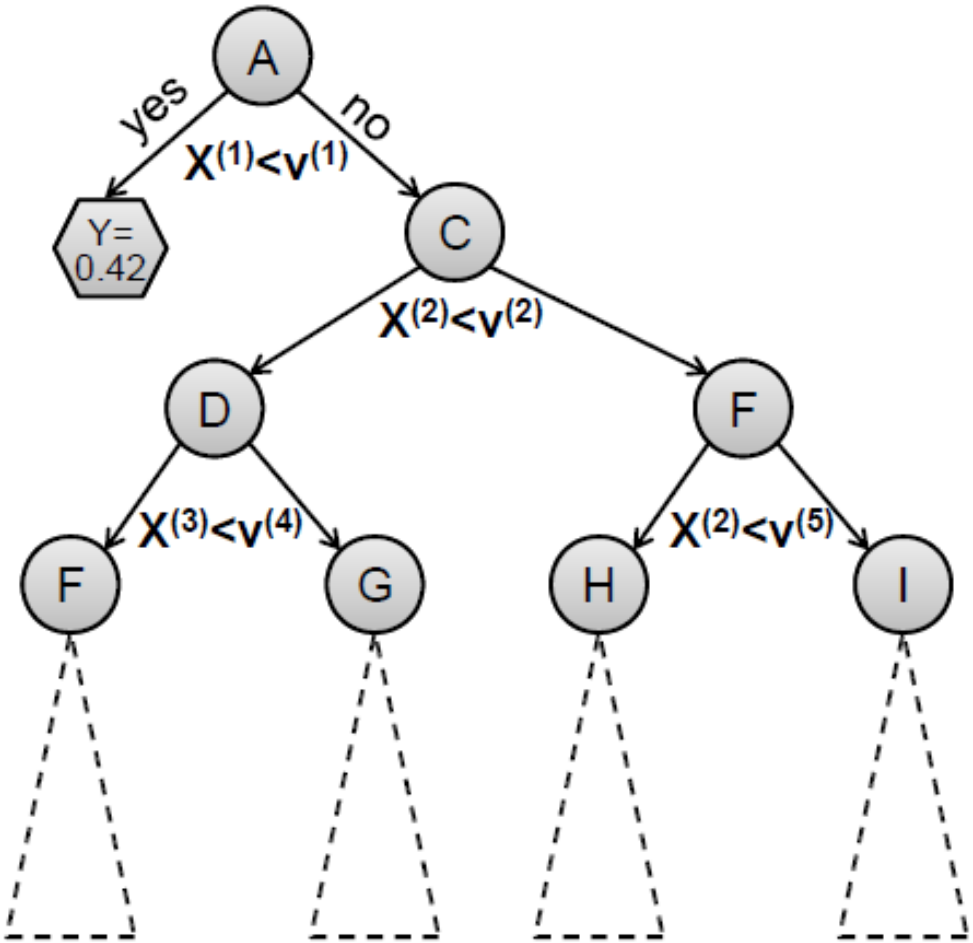
Decision Trees

- A **Decision Tree** is a tree-structured plan of a set of attributes to test in order to predict the output



Decision Trees (1)

- **Decision trees:**
  - Split the data at each internal node
  - Each leaf node makes a prediction
- **Lecture today:**
  - Binary splits: $\mathbf{X^{(j)}<v}$
  - Numerical attrs.
  - Regression

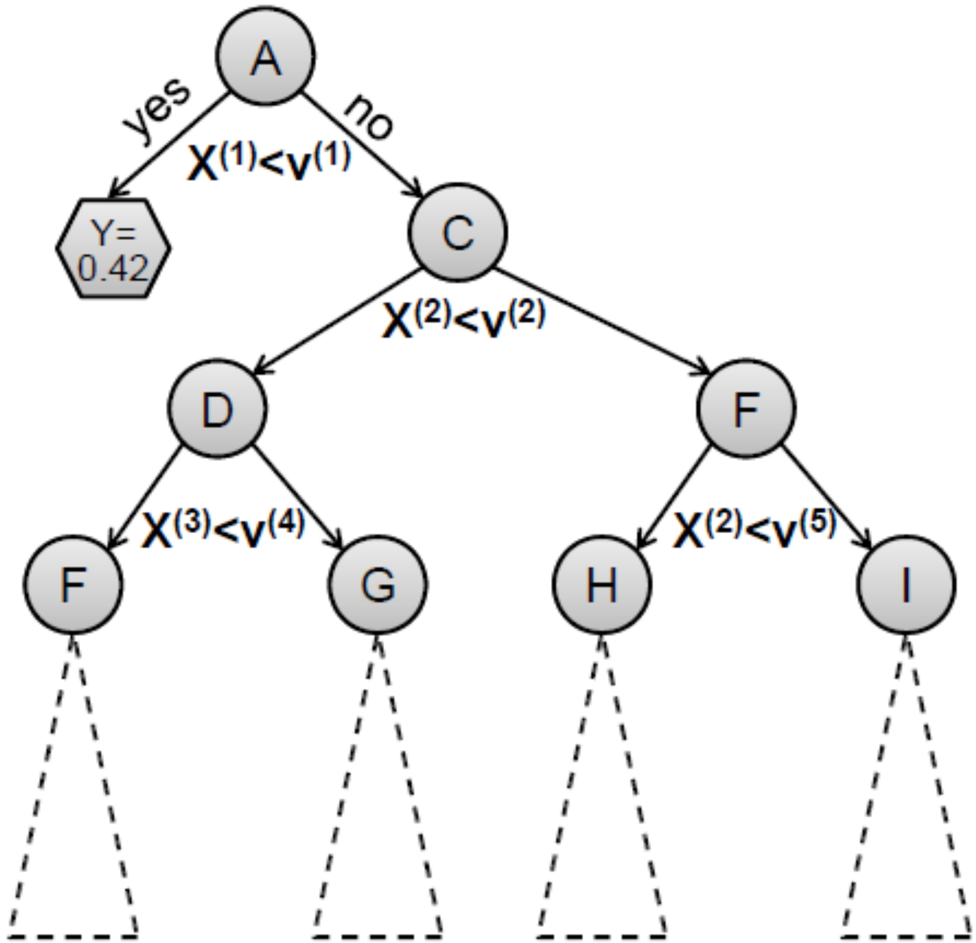


How to make predictions?

- **Input:** Example $\boldsymbol{x}_i$
- **Output:** Predicted $\boldsymbol{y}_i'$

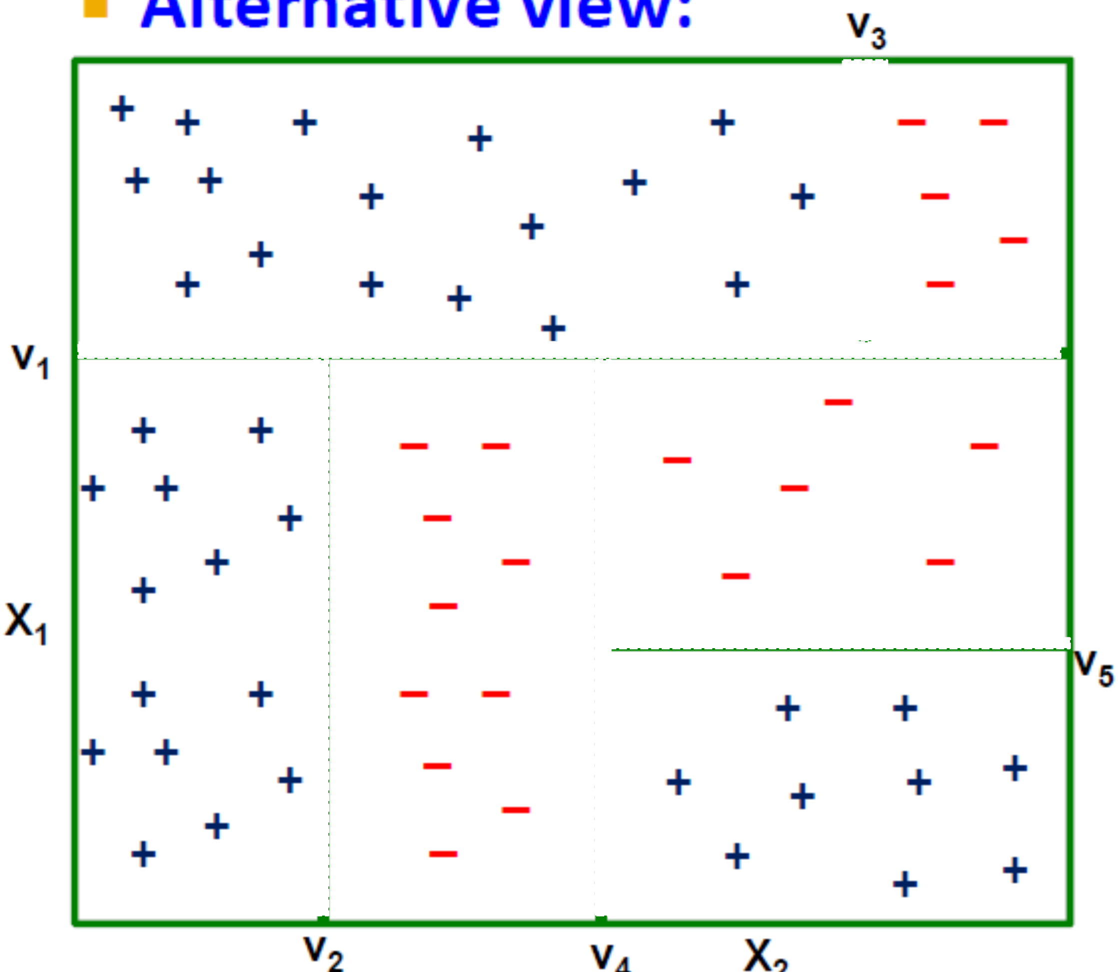
- "Drop" $\boldsymbol{x}_i$ down the tree until it hits a leaf node

- Predict the value stored in the leaf that $\boldsymbol{x}_i$ hits



Alternative View of a Tree

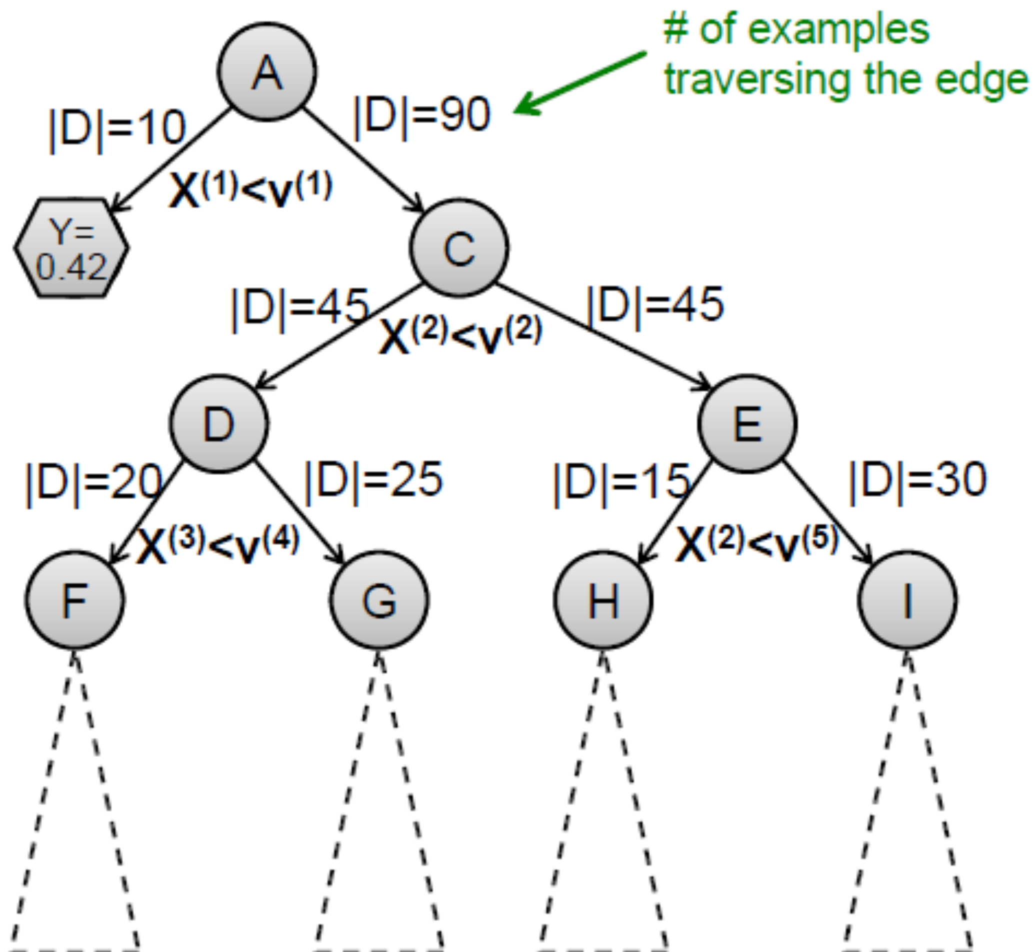
- **Alternative view:**



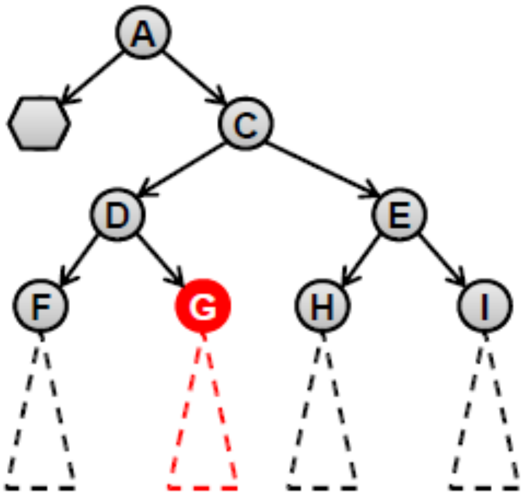
HOW TO CONSTRUCT A TREE

How to construct a tree?

- **Training dataset D*, |D*|=100 examples**



How to construct a tree?



- Imagine we are currently at some node G
  - Let D_G be the data that reaches G
- **There is a decision we have to make: Do we continue building the tree?**
  - **If yes**, which variable and which value do we use for a **split**?
    - Continue building the tree recursively
  - **If not**, how do we make a prediction?
    - We need to build a **"predictor node"**

How to construct a tree?

Algorithm 1 $\boxed{\textbf{BuildSubtree}}$

Require: Node n, Data $D \subset D^*$

1: $(n \rightarrow \text{split}, D_L, D_R) = \boxed{\text{FindBestSplit}(D)}$ **(1)**
2: **if** $\boxed{\text{StoppingCriteria}(D_L)}$ **then** **(2)**
3: $\quad n \rightarrow \text{left\_prediction} = \boxed{\text{FindPrediction}(D_L)}$ **(3)**
4: **else**
5: $\quad$ **BuildSubtree** $(n \rightarrow \text{left}, D_L)$
6: **if** StoppingCriteria(D_R) **then**
7: $\quad n \rightarrow \text{right\_prediction} = \text{FindPrediction}(D_R)$
8: **else**
9: $\quad$ **BuildSubtree** $(n \rightarrow \text{right}, D_R)$

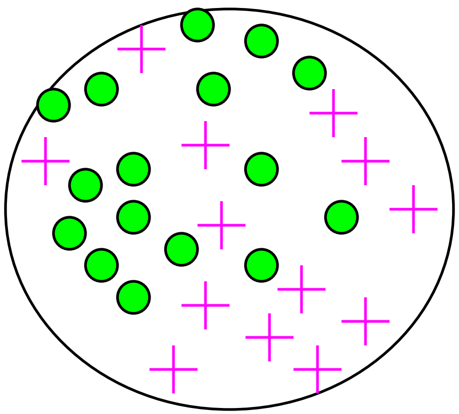
- **Requires at least a single pass over the data!**

Criterion for attribute selection

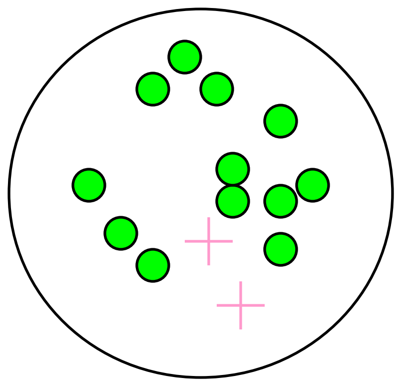
- Which is the best split?
  - The one which will result in the smallest tree
  - **Heuristic**: choose the attribute that produce s the "purest" nodes
- Need a good measure of purity!
  - Maximal when?
  - Minimal when?

Impurity

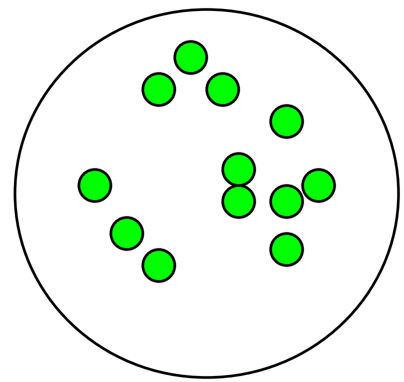
Very impure group



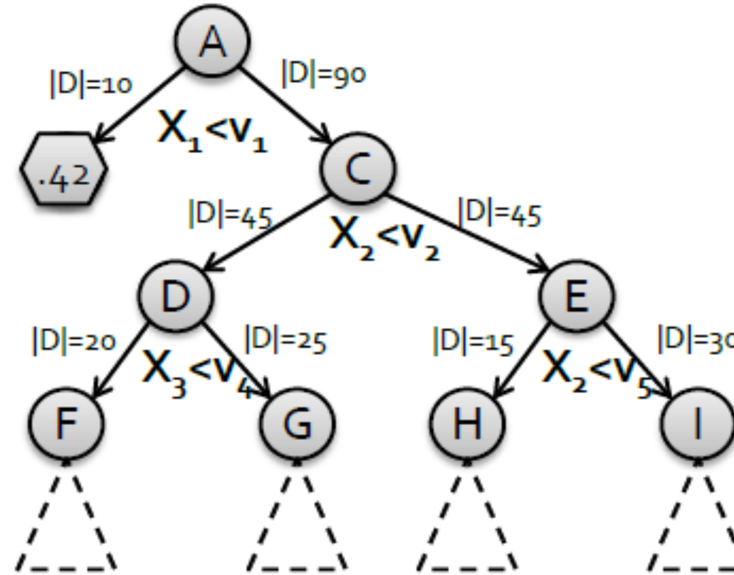
Less impure



Minimum impurity



How to construct a tree?

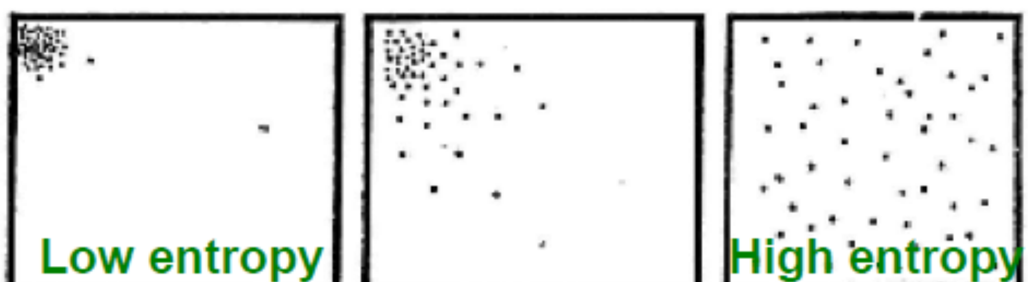


(1) How to split? Pick attribute & value that optimizes some criterion

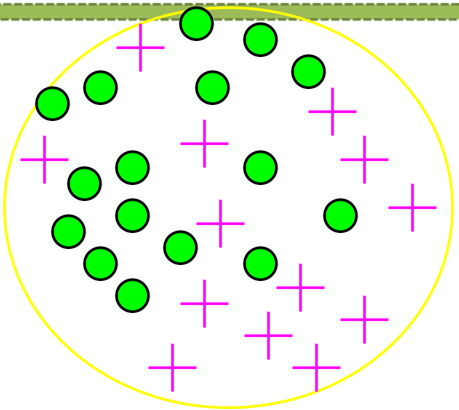
- **Classification:** **Information Gain**
  - **Measures how much a given attribute X tells us about the class Y**
  - **IG(Y | X)** : We must transmit **Y** over a binary link. How many bits on average would it save us if both ends of the line knew **X**?

Why Information Gain? Entropy

- **Entropy:** What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from **X**'s distribution?
- **The entropy of X:** $H(X) = -\sum_{j=1}^{m} p_j \log p_j$
  - "High Entropy": **X** is from a uniform (boring) distribution
    - A histogram of the frequency distribution of values of **X** is **flat**
  - "Low Entropy": **X** is from varied (peaks and valleys) distribution
    - A histogram of the frequency distribution of values of **X** would have many lows and one or two highs



Entropy: a common way to measure impurity



- Entropy = $\sum_i - p_i \log_2 p_i$

  p_i is the probability of class i
  Compute it as the proportion of class i in the set.

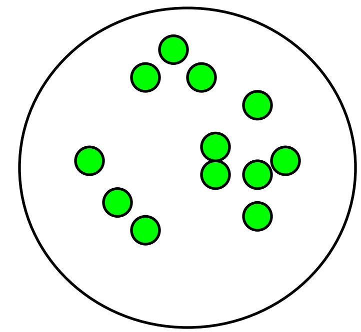
- Entropy comes from information theory. The higher the entropy the more the information content.

  What does that mean for learning from examples?

2-Class Cases:

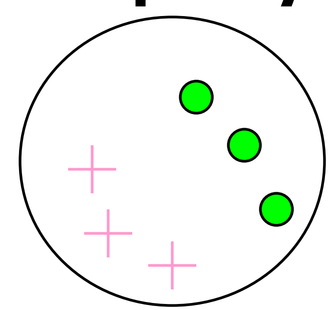
- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0

Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

Maximum impurity



Exercise

- Compute the entropy of the following distribution
  - { 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0 }

Exercise

- Compute the entropy of the following distribution
  - { blue, blue, red, orange, red, black, red, black, blue, blue, blue, blue, blue, red }

Information Gain

- We want to determine **which attribute** in a given set of training feature vectors is **most useful** for discriminating between the classes to be learned.

- **Information gain** tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Why Information Gain? Entropy

- **Suppose I want to predict 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
Math	Yes
History	No

- **From this data we estimate**
  - $P(Y = Yes) = 0.5$
  - $P(X = Math \,\&\, Y = No) = 0.25$
  - $P(X = Math) = 0.5$
  - $P(Y = Yes \mid X = History) = 0$
- **Note:**
  - $H(Y) = -½\log_2(½) - ½\log_2(½) = \mathbf{1}$
  - $H(X) = 1.5$

Why Information Gain? Entropy

- **Suppose I want to predict 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
Math	Yes
History	No

- **Def: Specific Conditional Entropy**
  - **H(Y | X=v)** = The entropy of **Y** among only those records in which **X** has value **v**
  - **Example:**
    - $H(Y|X = Math) = 1$
    - $H(Y|X = History) = 0$
    - $H(Y|X = CS) = 0$

Why Information Gain?

- **Suppose I want to predict 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
Math	Yes
History	No

- **Def: Conditional Entropy**
  - $H(Y \mid X)$ = 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**
  - $= \sum_j P(X = v_j) H(Y|X = v_j)$

Why Information Gain?

- **Suppose I want to predict Y and I have input X**

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

  - $H(Y \mid X)$ = The average specific conditional entropy of Y

$$= \sum_j P(X = v_j) H(Y|X = v_j)$$

  - **Example:**

V_j	$P(X=v_j)$	$H(Y\|X=v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

  - **So:** H(Y|X)=0.5*1+0.25*0+0.25*0 = **0.5**

Why Information Gain?

- Suppose I want to predict $\boldsymbol{Y}$ and I have input $\boldsymbol{X}$

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
Math	Yes
History	No

- **Def: Information Gain**
  - $\boldsymbol{IG(Y|X)}$ = I must transmit **Y**. **How many bits on average would it save me if both ends of the line knew X?**

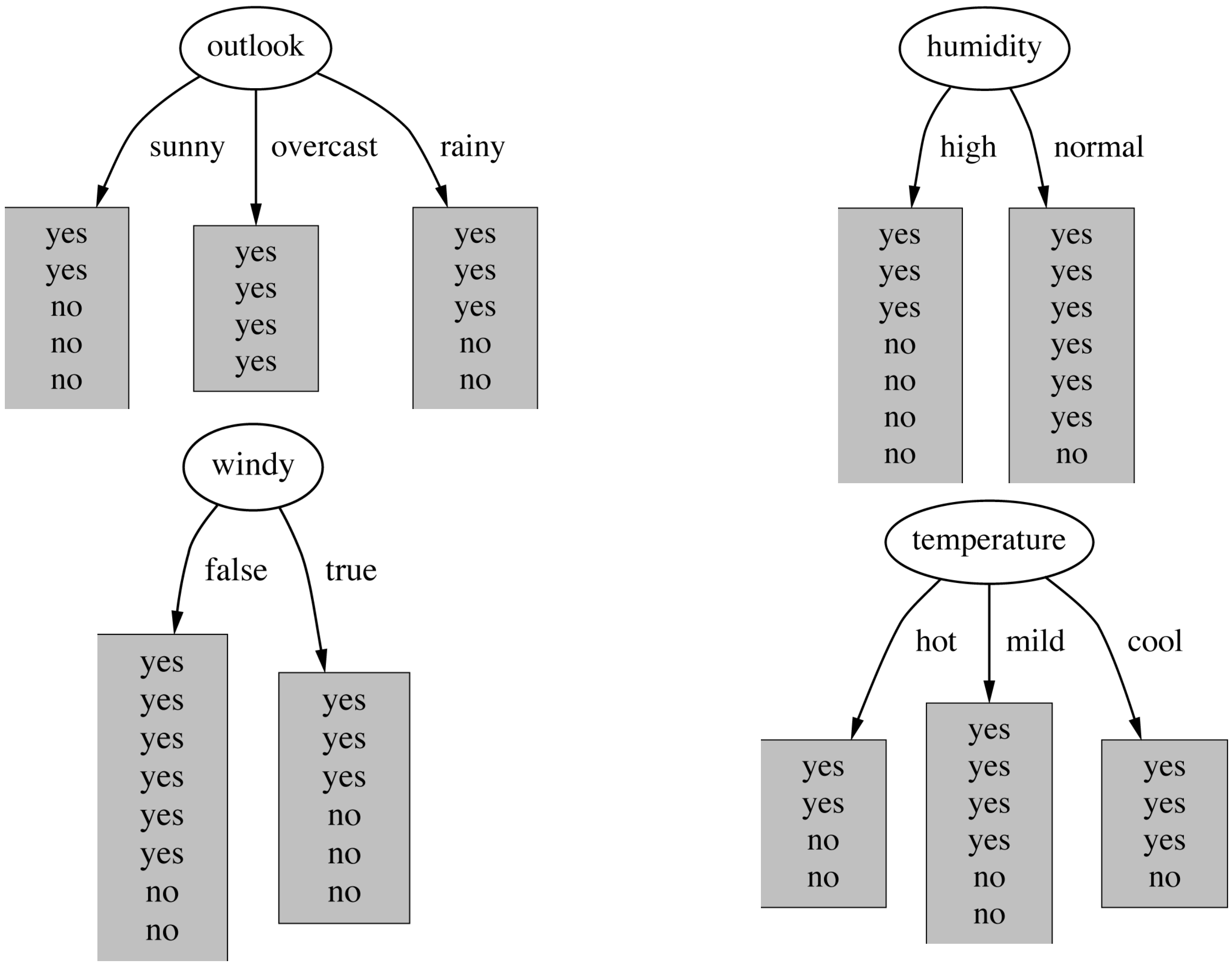
$$\boldsymbol{IG(Y|X) = H(Y) - H(Y \mid X)}$$

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

What is Information Gain used for?

- **Suppose you are trying to predict whether someone is going live past 80 years**
- From historical data you might find:
  - **IG(LongLife | HairColor) = 0.01**
  - **IG(LongLife | Smoker) = 0.3**
  - **IG(LongLife | Gender) = 0.25**
  - **IG(LongLife | LastDigitOfSSN) = 0.00001**
- **IG tells us how much information about Y is contained in X**
  - **So attribute X that has high IG(Y|X) is a good split!**

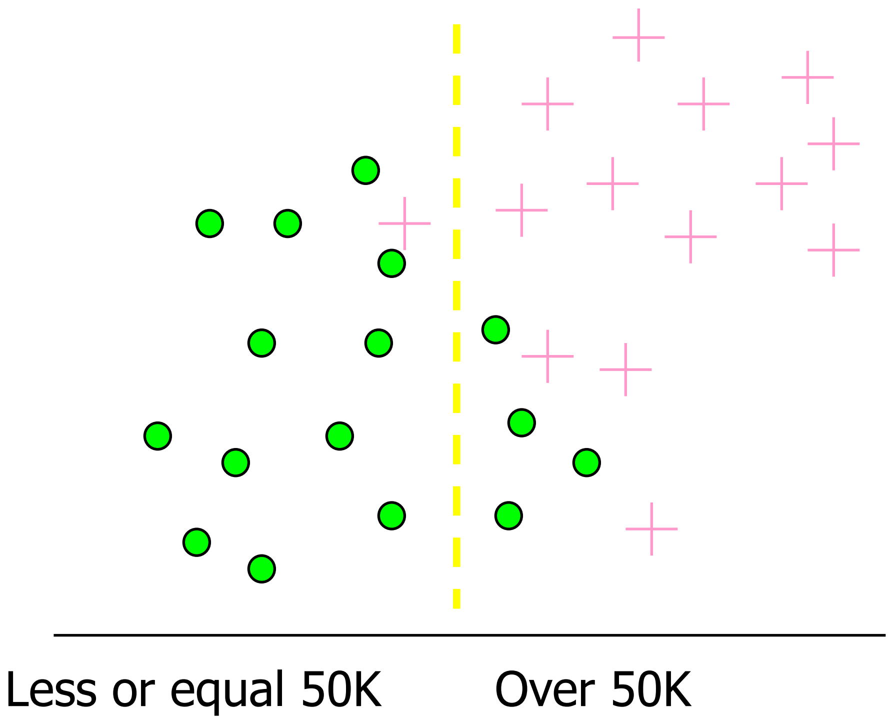
Which attribute to select?



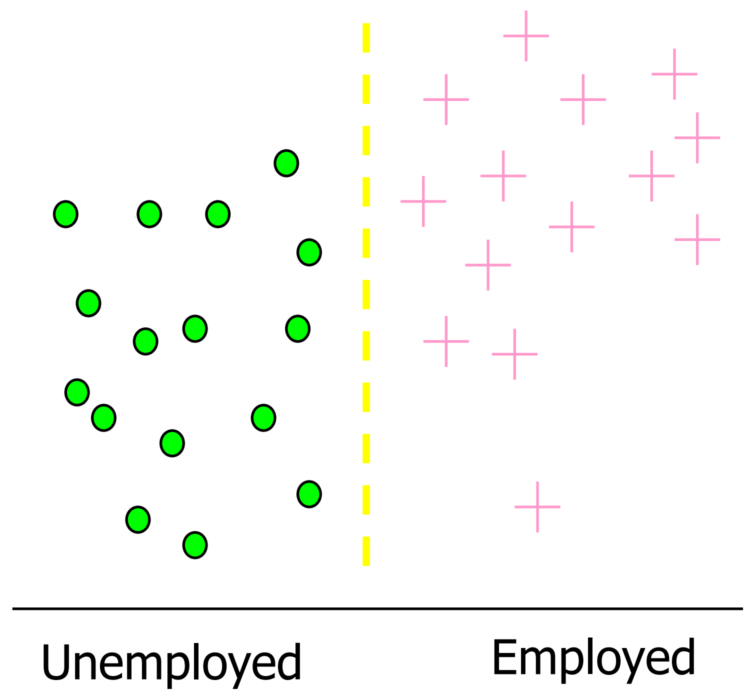
Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K



Split over whether applicant is employed

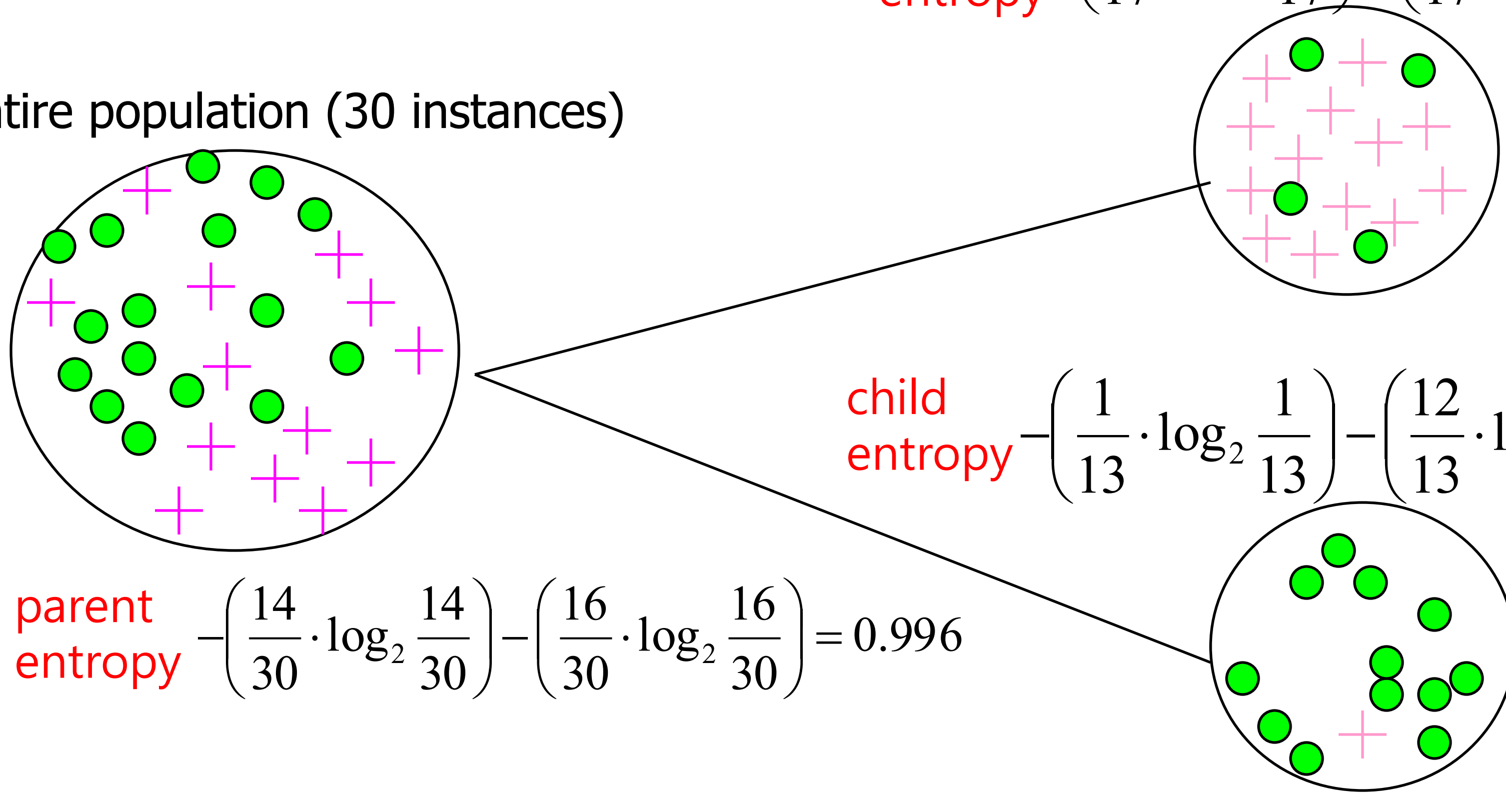


Exercise

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



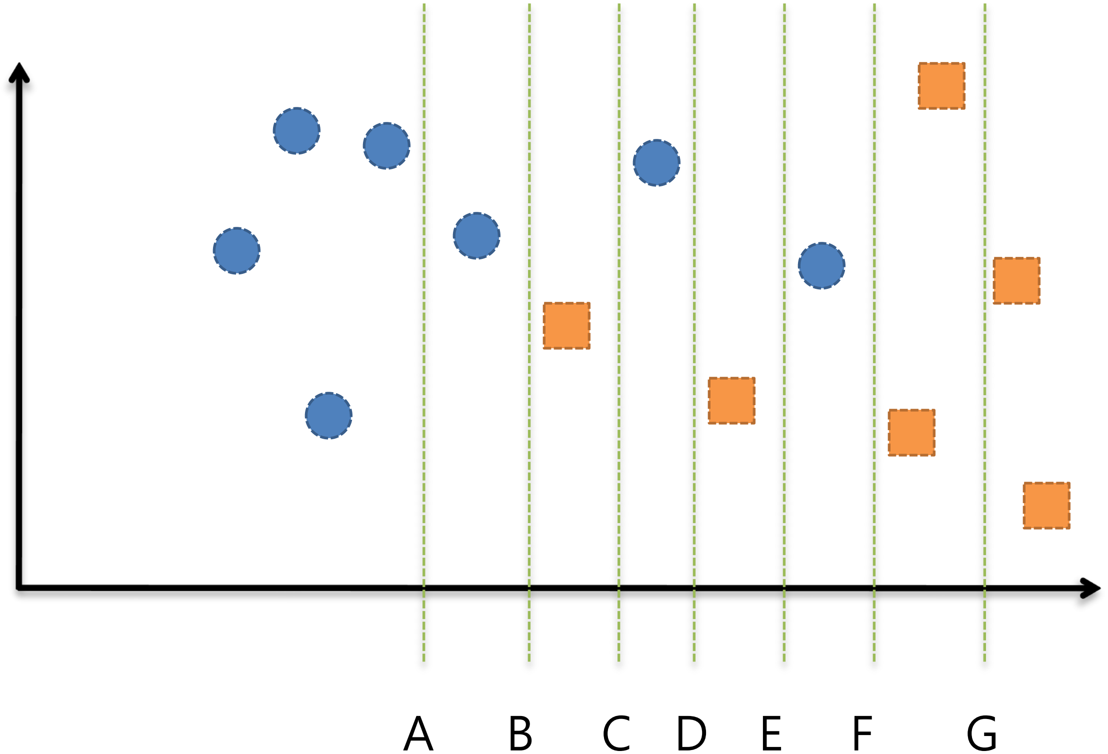
child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Exercise



Which one is the best split point?

DECISION TREE BUILDING ALGORITHM

Decision Tree Algorithm

- A decision tree is created in two phases:
  - Building Phase
    - Recursively split nodes using best splitting attribute for node until all the examples in each node belong to one class
  - Pruning Phase
    - Prune leaf nodes recursively to prevent over-fitting
    - Smaller imperfect decision tree generally achieves better accuracy

Building Phase

- General tree-growth algorithm (binary tree)
  - Build a tree recursively
- <u>**Partition**</u>(Data S)
-         If (all points in S are of the same class) then return;
-         for each attribute A do
-                 evaluate splits on attribute A;
-         Use **best split** to partition S into S1 and S2;
-         *Partition*(S1);
-         *Partition*(S2);

Summary

How to construct a tree?

Algorithm 1 **BuildSubtree**

Require: Node n, Data $D \subset D^*$

$(n \rightarrow \text{split}, D_L, D_R) = \text{FindBestSplit}(D)$ **(1)**
if $\text{StoppingCriteria}(D_L)$ **then** **(2)**
    $n \rightarrow \text{left\_prediction} = \text{FindPrediction}(D_L)$ **(3)**
else
    BuildSubtree $(n \rightarrow \text{left}, D_L)$
if $\text{StoppingCriteria}(D_R)$ **then**
    $n \rightarrow \text{right\_prediction} = \text{FindPrediction}(D_R)$
else
    BuildSubtree $(n \rightarrow \text{right}, D_R)$

- **Requires at least a single pass over the data!**