Lecture 10: Decision Trees

Artificial Intelligence
CS-UY-4613-A / CS-GY-6613-I

Julian Togelius
julian.togelius@nyu.edu

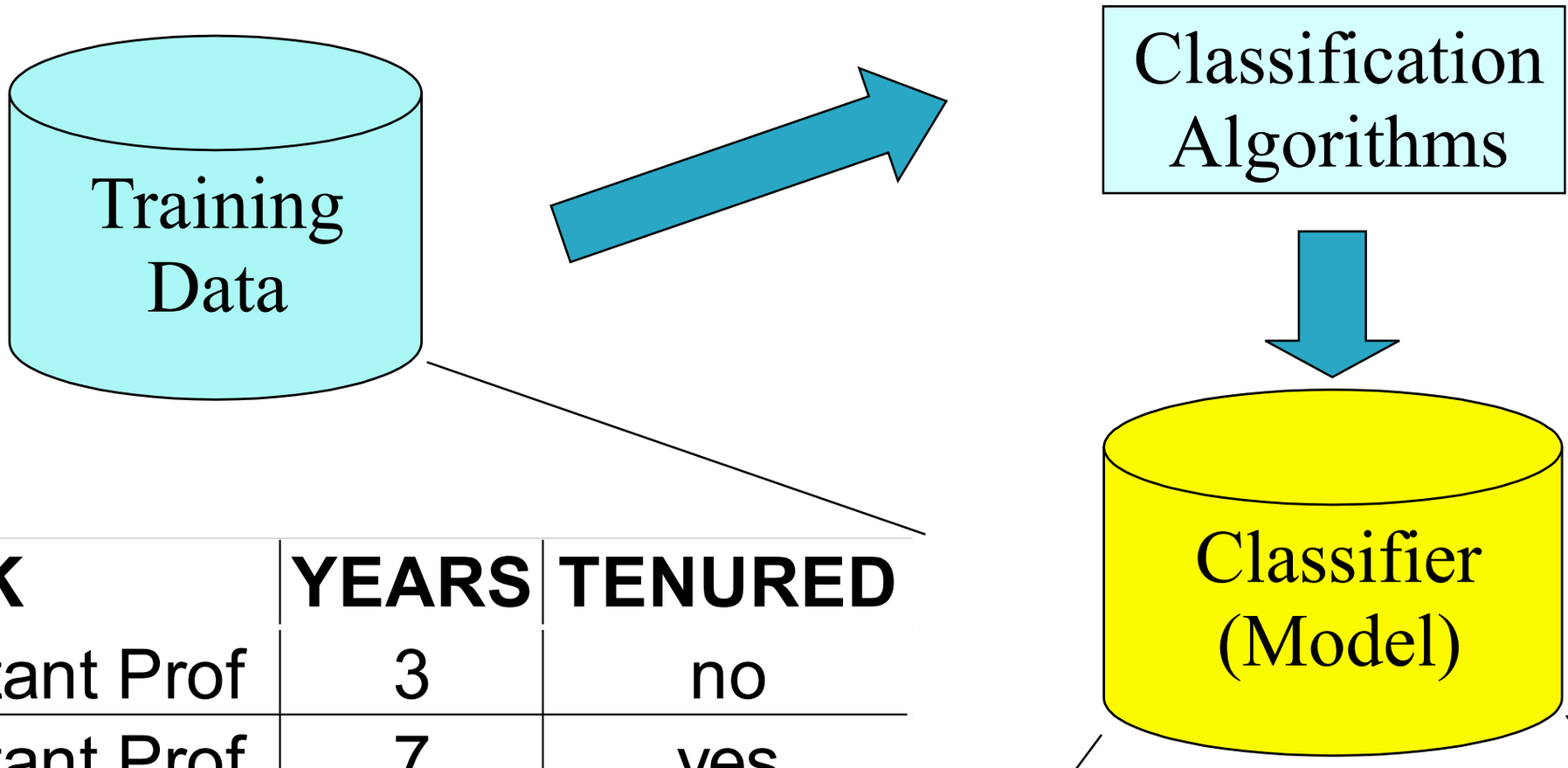
Types of learning

- **Supervised learning**
  Learning to predict or classify labels based on labeled input data

- **Unsupervised learning**
  Finding patterns in unlabeled data

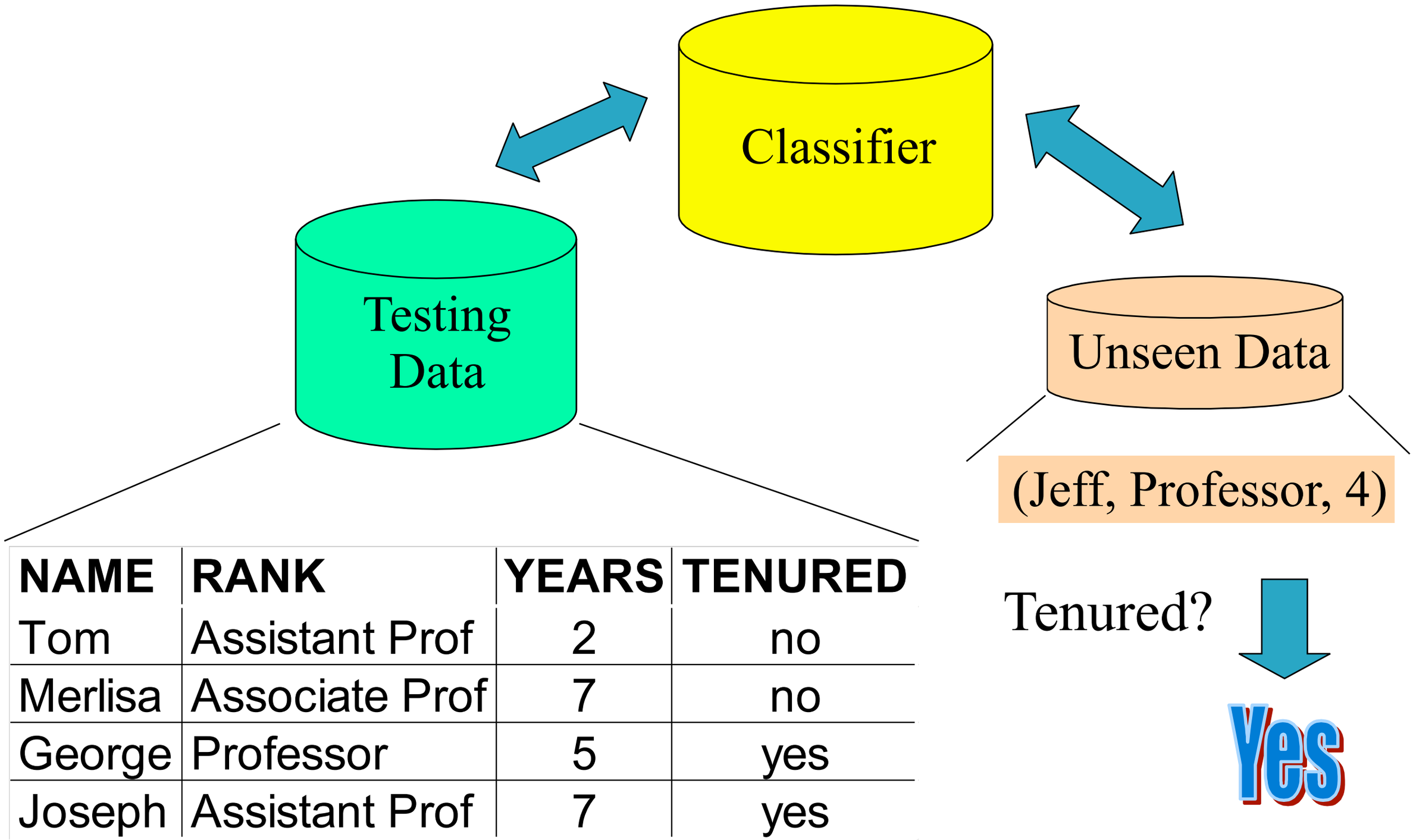
- **Reinforcement learning**
  Learning well-performing behavior from state observations and rewards

Model construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Using the model



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classification vs prediction

- Classification: binary or nominal labels
    - Examples: pregnant or not, from which country, which type of road sign
- Prediction: continous labels
    - Examples: future stock price, life expectancy, distance to obstacle

Terminology (supervised learning)

- Each line of data: instance / data point / tuple
- The features of each instance: features / attributes
- That which should be learned: labels / targets
- Each instance has features and a label
- We train on the training set…
- …and test on the testing set

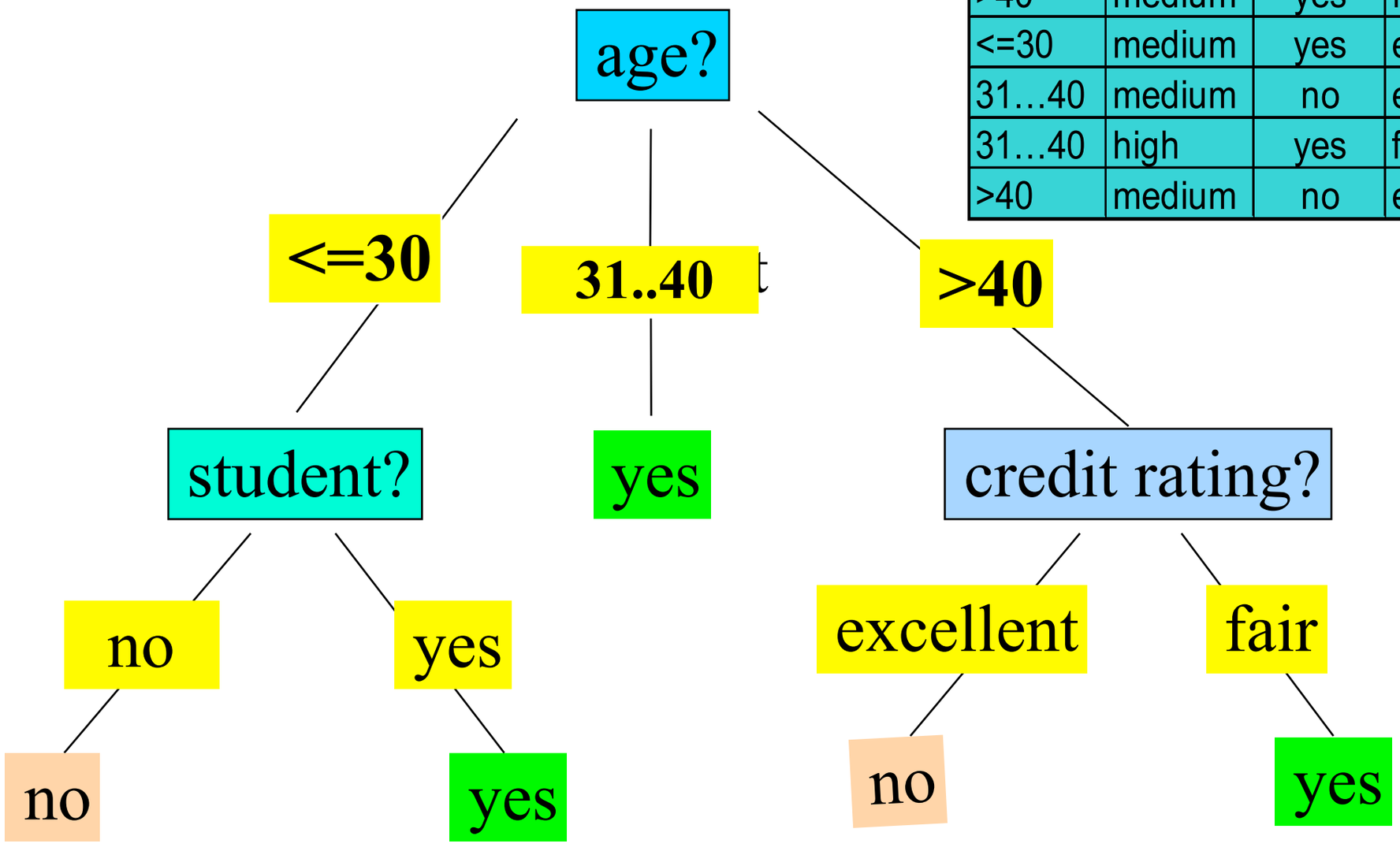
Decision trees

- A popular representation for classifiers
- Human-readable
- Can be learned (induced) efficiently using algorithms based on information theory
  - An eager learning method
- Often yields high-accuracy classifiers

An example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Classify: buys_computer



ID3 Algorithm for Decision Tree Induction

- Tree is constructed in a top-down recursive divide-and-conquer manner

- At start, all the training examples are at the root

- Attributes are categorical (if continuous-valued, they are discretized in advance)

- Examples are partitioned recursively based on selected attributes

- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

ID3 Algorithm for Decision Tree Induction

- Stop partioning when...
  - All samples for a given node belong to the same class, or…
  - There are no remaining attributes for further partitioning – (vote on the leaf) or…
  - There are no samples left

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class Ci, estimated by $|C_i, D|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

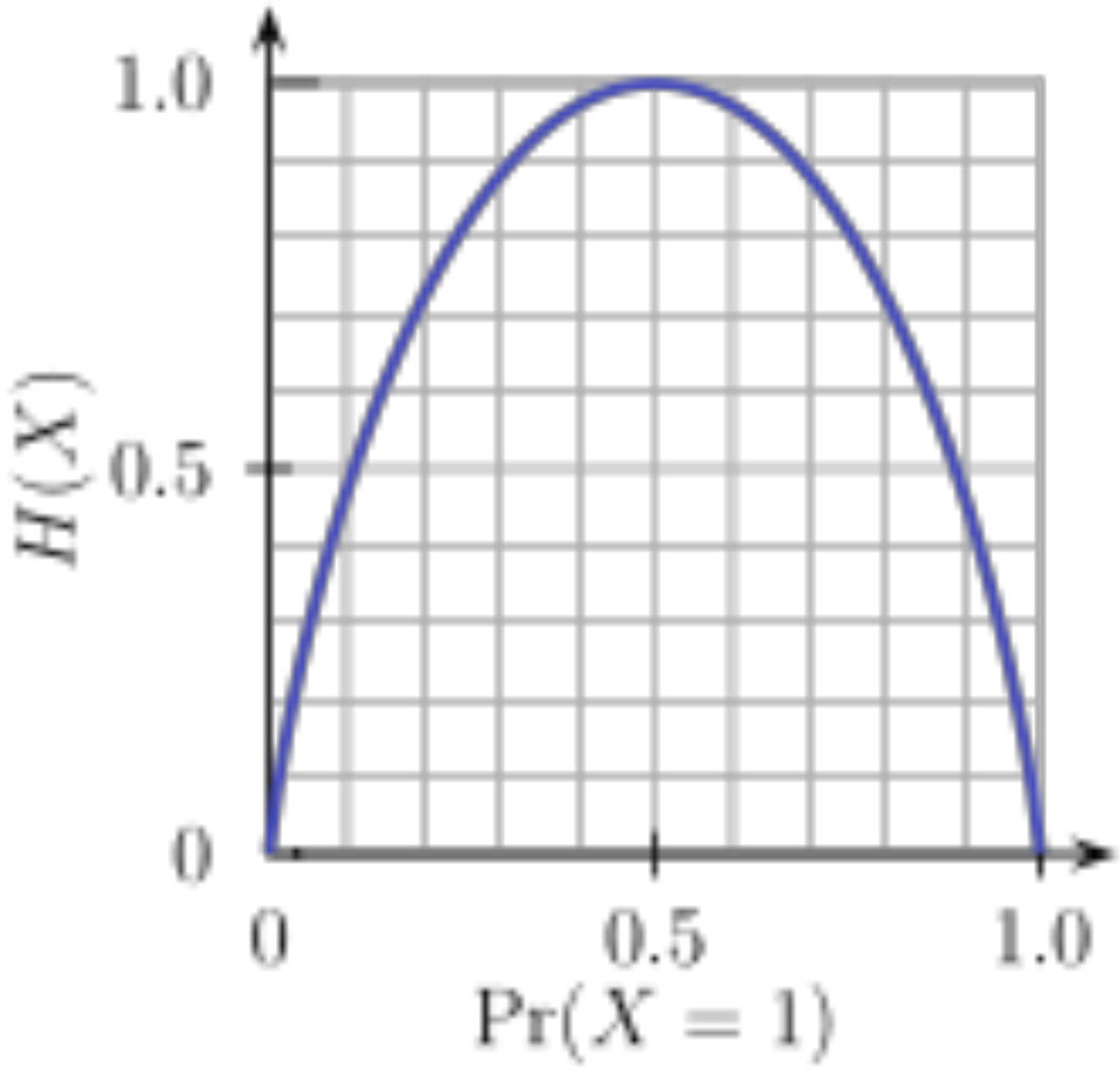
- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Binary entropy function



Attribute Selection: Information Gain

Class P: buys_computer = “yes”

Class N: buys_computer = “no”

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means “age <=30” has 5 out of 14 samples, with 2 yes’es and 3 no’s. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

- Input: A data set, S
  Output: A decision tree
    - If all the instances have the same value for the target attribute then return a decision tree that is simply this value (not really a tree - more of a stump).
- Else
    - Compute Gain values (see above) for all attributes and select an attribute with the highest value and create a node for that attribute.
    - Make a branch from this node for every value of the attribute
    - Assign all possible values of the attribute to branches.
    - Follow each branch by partitioning the dataset to be only instances whereby the value of the branch is present and then go back to 1.

function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** tree

  if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same classification **then return** the classification
  else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
  else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples \textbf{ and } e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

Overfitting and pruning

- Overfitting: An induced tree may overfit the training data (specific instance of a concept that applies to all supervised learning)
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early– do not split a node if this would result in the goodness measure (e.g. accuracy) falling below a threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Further notes on project

- You ALWAYS search in state space, i.e. always use a forward model to simulate actions rather than just moving around on the map

- Depth-first and breadth first are likely to work very badly on any game in their naive form

- It's up to up to you whether to "fix" this so the algorithms do well

  - Depth-limited search, e.g. 10

  - Use an evaluation function, e.g. score