Machine Learning

CS 539

Worcester Polytechnic Institute
Department of Computer Science
Instructor: Prof. Kyumin Lee

Project Description & Examples

- Project Description
  - https://canvas.wpi.edu/courses/58900/pages/project-description

- Previous years' projects
  - https://gabrieldeml.github.io/CS539-final/
  - https://sites.google.com/view/cs539finalproject/home
  - https://sites.google.com/view/cs539domain/project-model
  - https://bl75d.github.io/CS539MLFinalProj/
  - https://sites.google.com/view/cs539project-h1bcaseprediction/home
  - https://sites.google.com/view/audio-genre-classification/home
  - https://r3av3r97.github.io/FinalProject_ML_CS539/

How to find interesting research papers?

- Search research papers via Google Scholar or visit well-known conference/journal sites to see what papers have been recently published.
- Here are some of top conferences related to Machin Learning and its application:
  - CVPR (IEEE Conference on Computer Vision and Pattern Recognition), NeurIPS (Neural Information Processing Systems), ICML (International Conference on Machine Learning), EMNLP (Conference on Empirical Methods in Natural Language Processing), ICLR (International Conference on Learning Representations), AAAI (AAAI Conference on Artificial Intelligence), ICCV (IEEE International Conference on Computer Vision), SIGKDD (ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), ACL (Meetings of the Association for Computational Linguistics), NAACL (Annual Conference of the North American Chapter of the Association for Computational Linguistics), SIGIR (Special Interest Group on Information Retrieval), etc.

How to find interesting research papers?

- Encourage you to read full/long papers each of which usually have **at least 8 pages in double column or 12 pages in one column**

- When you search a certain conference, add the conference acronym with one of recent years**:**
  - For example, search ICML 2023 or ICML 2022 and visit its conference page and see a list of published papers.

- How to Read a CS Research Paper

Types of Learning

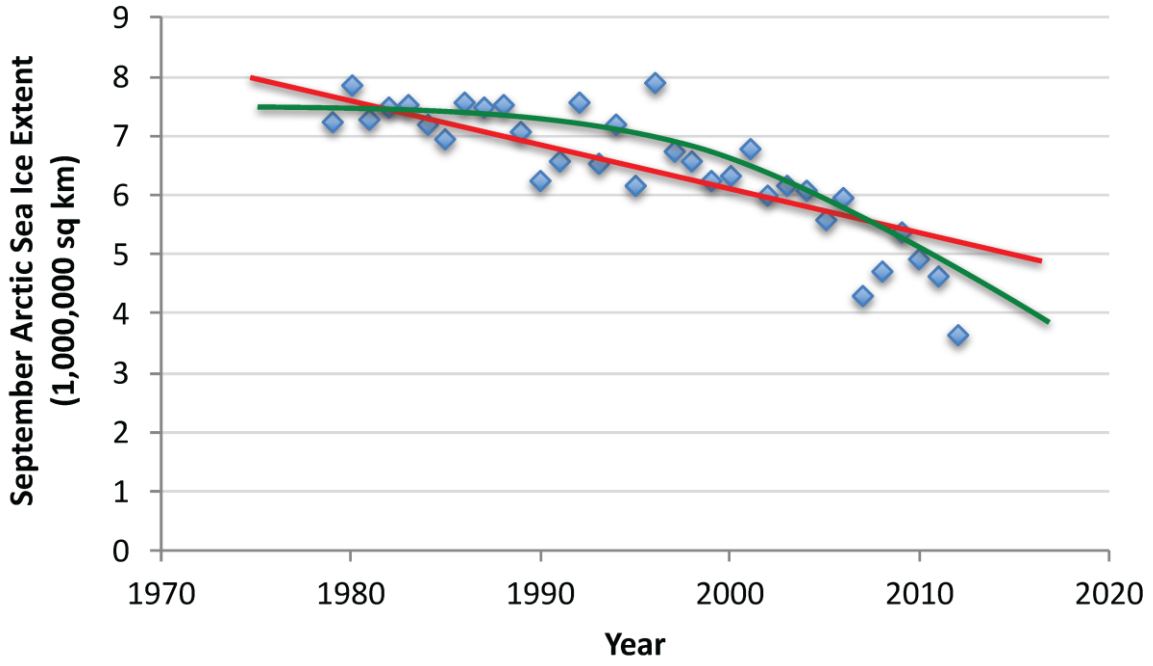
Types of Learning

- **Three Types of Major Learning**
  - **Supervised (inductive) learning**
    - Given: training data + desired outputs (labels)
  - **Unsupervised learning**
    - Given: training data (without desired outputs)
  - **Reinforcement learning**
    - Rewards from sequence of actions

- **Other Types of Learning**
  - **Semi-supervised learning**
    - Given: training data + a few desired outputs
  - **Transfer Learning**
    - A model, which is developed for a task, is reused as the starting point for a model on a second task
      - E.g., Resnet, BERT

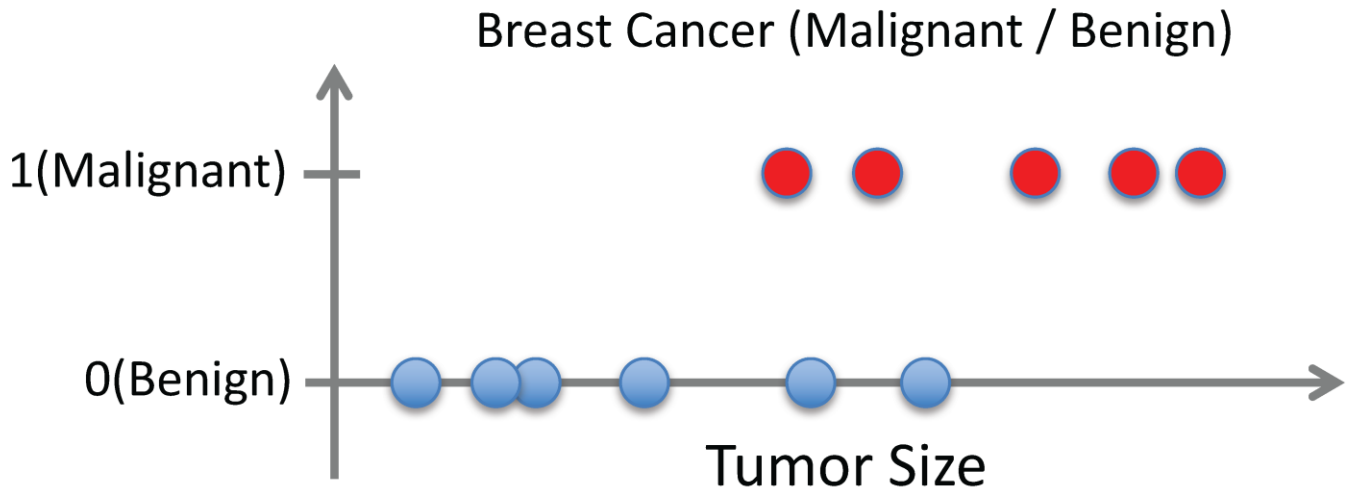
Supervised Learning: Regression

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is real-valued == regression



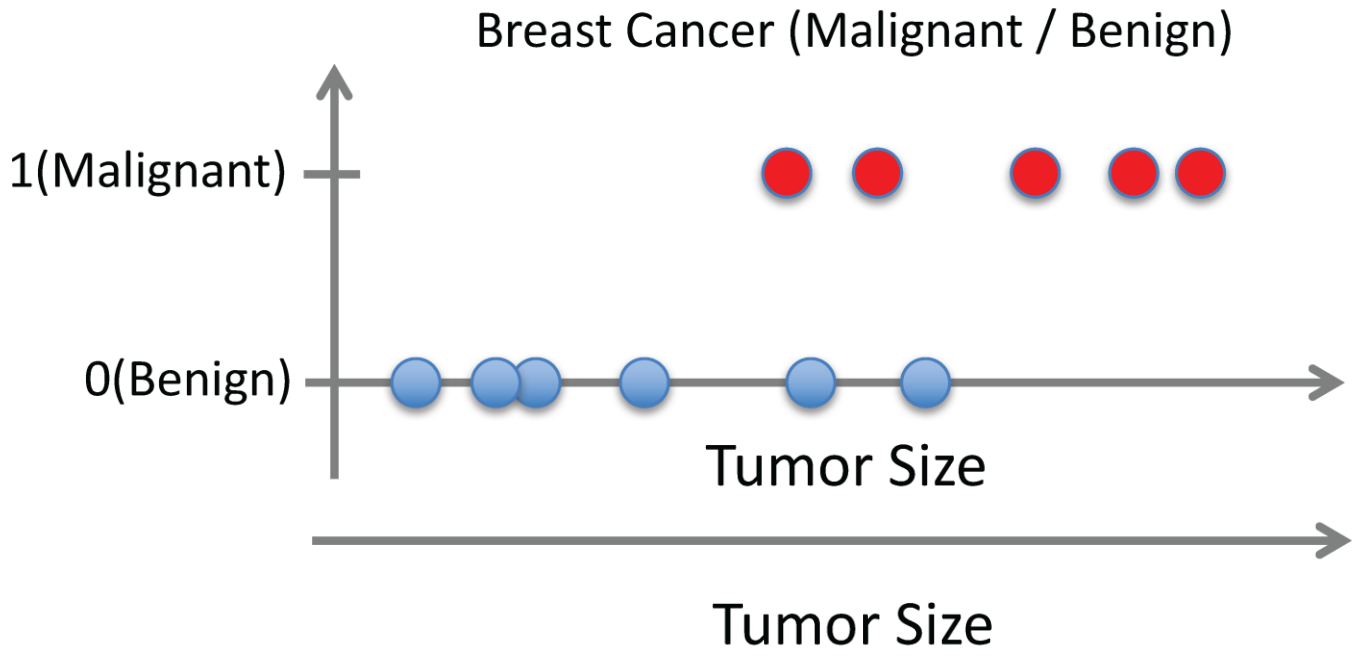
Supervised Learning: Classification

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



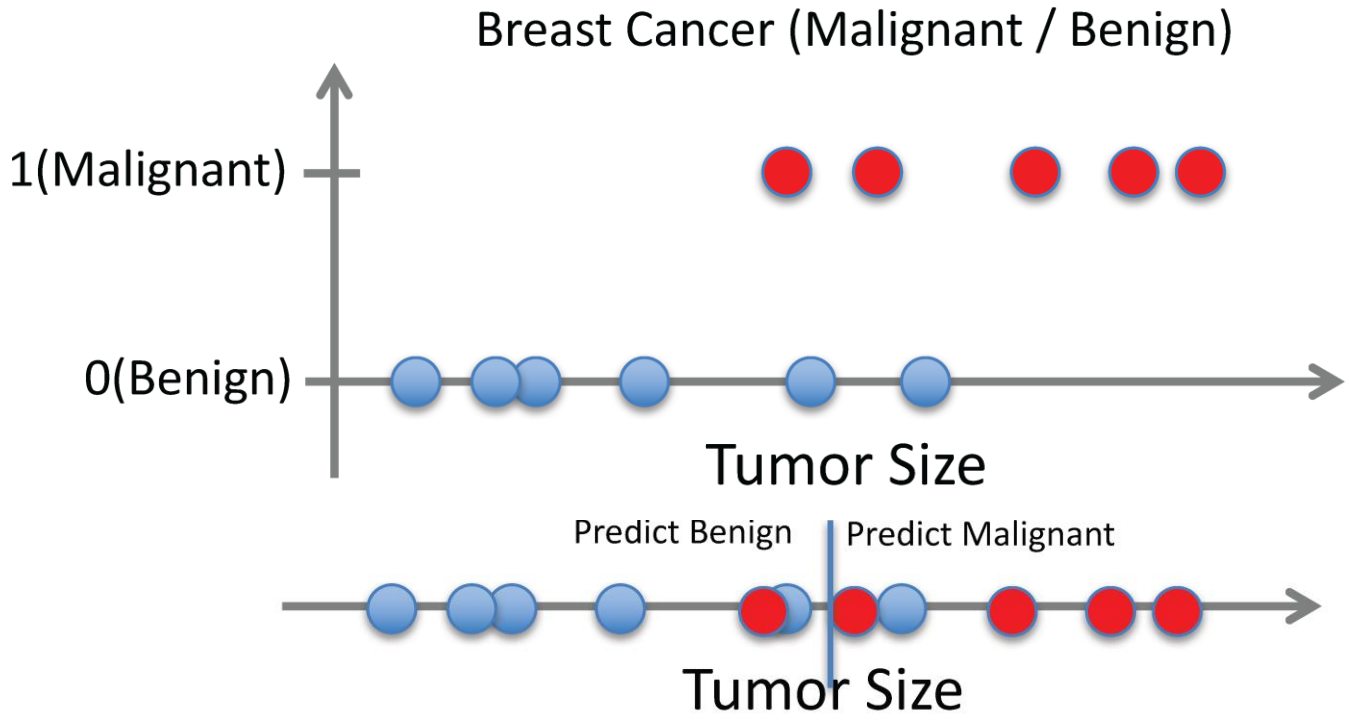
Supervised Learning: Classification

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



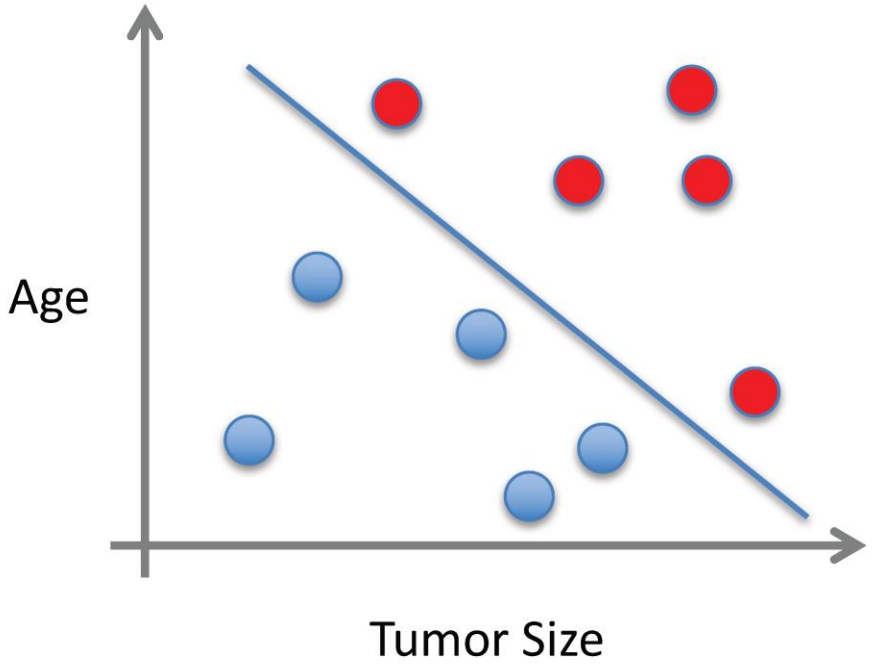
Supervised Learning: Classification

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



Supervised Learning

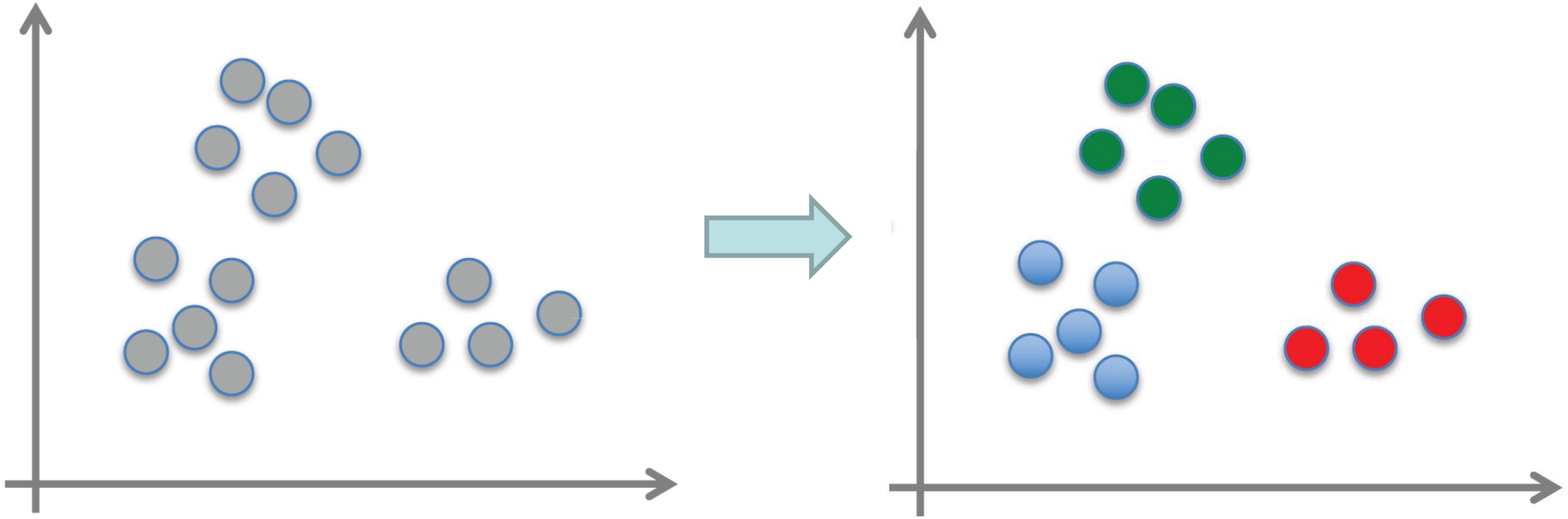
- x can be multi-dimensional
  - Each dimension corresponds to an attribute



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
…

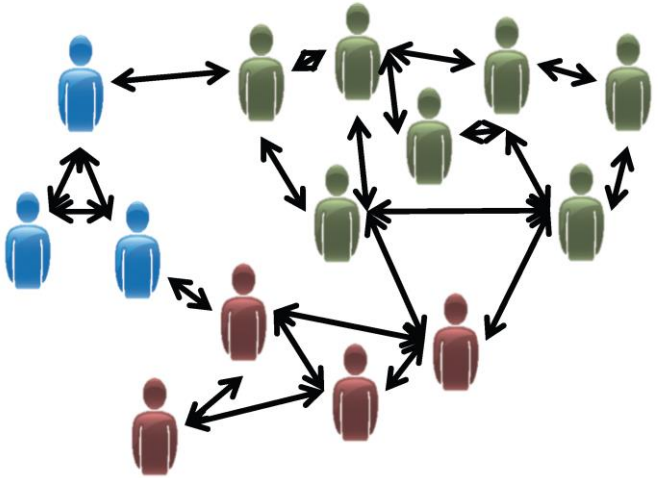
Unsupervised Learning

- Given $x_1, x_2, \ldots, x_n$ (without labels)
- Output hidden structure behind the x's
  - E.g., clustering



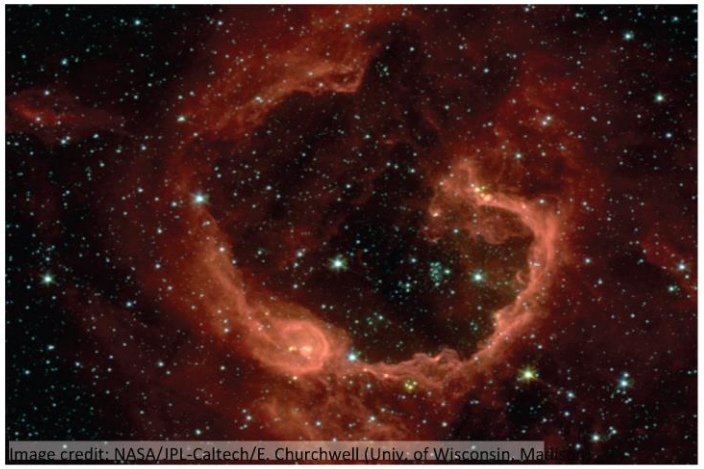
Unsupervised Learning

Organize computing clusters



Social network analysis

Market segmentation

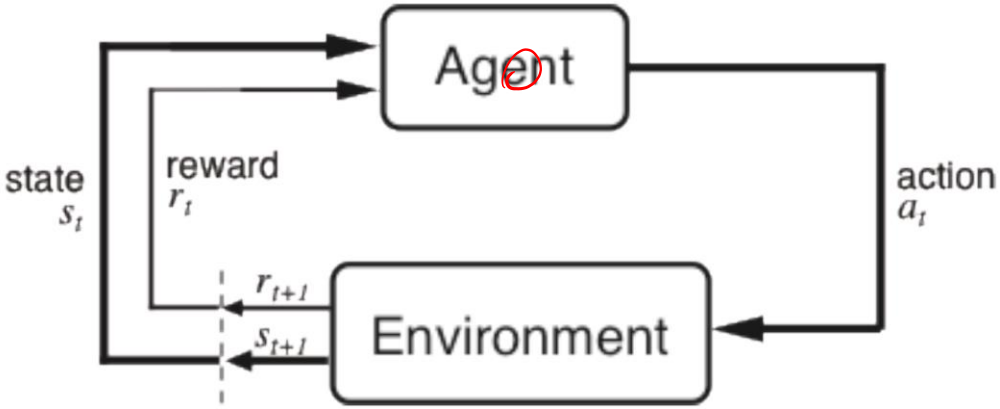


Astronomical data analysis

Reinforcement Learning

- Given a sequence of states and actions with (delayed) rewards, output a policy
  - Policy is a mapping from states → actions that tells you what to do in a given state
- Examples:
  - Game playing (e.g., Go!)
  - Robot in a maze
  - Balance a pole on your hand

The Agent-Environment Interface



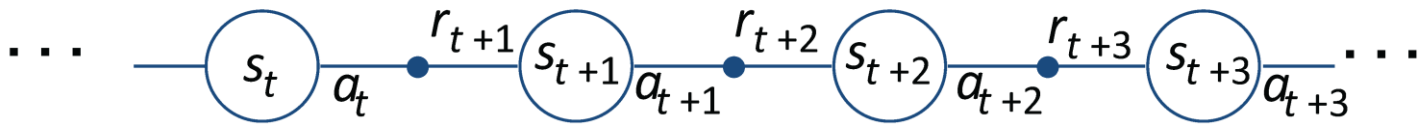
Agent and environment interact at discrete time steps : $t = 0, 1, 2, \mathrm{K}$

Agent observes state at step t: $s_t \in S$

produces action at step t: $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \Re$

and resulting next state : s_{t+1}



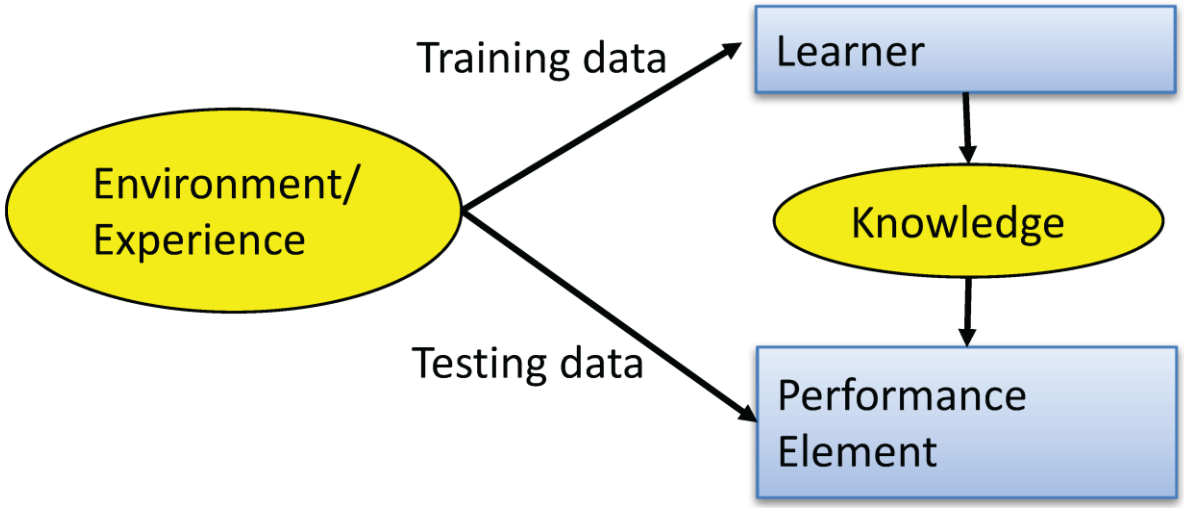
Reinforcement Learning

https://www.youtube.com/watch?v=4cgWya-wjgY

Framing a Learning Problem

Designing a Learning System

- Choose the training experience
- Choose exactly what is to be learned
  - i.e. the *target function*
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience



Training vs. Test Distribution

- We generally assume that the training and test examples are independently drawn from the same overall distribution of data
  - We call this “i.i.d” which stands for “independent and identically distributed”

ML in a Nutshell

- Tens of thousands of machine learning algorithms
  - Hundreds new every year

- Every ML algorithm has three components:
  - **Representation**
  - **Optimization**
  - **Evaluation**

Various Function Representations

- Numerical functions
  - Linear regression
  - Neural networks
  - Support vector machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic Graphical Models
  - Naive Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMMs)
  - Probabilistic Context Free Grammars (PCFGs)
  - Markov networks

Various Search/Optimization Algorithms

- Gradient descent
  - Perceptron
  - Backpropagation
- Dynamic Programming
  - HMM Learning
  - PCFG Learning
- Divide and Conquer
  - Decision tree induction
  - Rule learning
- Evolutionary Computation
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)
  - Neuro-evolution

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- etc

ML in Practice

- Understand domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learn models
- Interpret results
- Consolidate and deploy discovered knowledge

Lessons Learned about Learning

- Learning can be viewed as using direct or indirect experience to approximate a chosen target function.
- Function approximation can be viewed as a search through a space of hypotheses (representations of functions) for one that best fits a set of training data.
- Different learning methods assume different hypothesis spaces (representation languages) and/or employ different search techniques.

A Brief History of Machine Learning

History of Machine Learning

- 1950s
  - Samuel's checker player
  - Selfridge's Pandemonium
- 1960s:
  - Neural networks: Perceptron
  - Pattern recognition
  - Learning in the limit theory
  - Minsky and Papert prove limitations of Perceptron
- 1970s:
  - Symbolic concept induction
  - Winston's arch learner
  - Expert systems and the knowledge acquisition bottleneck
  - Quinlan's ID3
  - Michalski's AQ and soybean diagnosis
  - Scientific discovery with BACON
  - Mathematical discovery with AM

History of Machine Learning

- 1980s:
  - Advanced decision tree and rule learning
  - Explanation-based Learning (EBL)
  - Learning and planning and problem solving
  - tility problem
  - Analogy
  - Cognitive architectures
  - Resurgence of neural networks (connectionism, backpropagation)
  - Valiant’s PAC Learning Theory
  - Focus on experimental methodology
- 1990s
  - Data mining
  - Adaptive software agents and web applications
  - Text learning
  - Reinforcement learning (RL)
  - Inductive Logic Programming (ILP)
  - Ensembles: Bagging, Boosting, and Stacking
  - Bayes Net learning

History of Machine Learning

- 2000s
  - Support vector machines & kernel methods
  - Graphical models
  - Statistical relational learning
  - Transfer learning
  - Sequence labeling
  - Collective classification and structured outputs
  - Computer Systems Applications (Compilers, Debugging, Graphics, Security)
  - E-mail management
  - Personalized assistants that learn
  - Learning in robotics and vision

- 2010s and present
  - Deep learning systems
  - Learning for big data
  - Bayesian methods
  - Multi-task & lifelong learning
  - Applications to vision, speech, social networks, learning to read, etc.
  - Multimodal machine learning
  - Generative AI
  - ???

What We'll Cover in this Course

- **Supervised learning**
  - Linear regression
  - Logistic regression
  - Support vector machines & kernel methods
  - Model ensembles
  - Neural networks & deep learning
  - Learning theory
- **Unsupervised learning**
  - Clustering
  - Dimensionality reduction
- **Reinforcement learning**
- **Transfer learning**
- **Evaluation**
- **Applications**

Decision Trees

Function Approximation

- Problem Setting
  - Set of possible instances $\mathcal{X}$
  - Set of possible labels $\mathcal{Y}$
  - Unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$
  - Set of function hypotheses $H = \{ h \mid h : \mathcal{X} \rightarrow \mathcal{Y} \}$

- Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

- Output: Hypothesis $h \in H$ that best approximates f

Sample Dataset

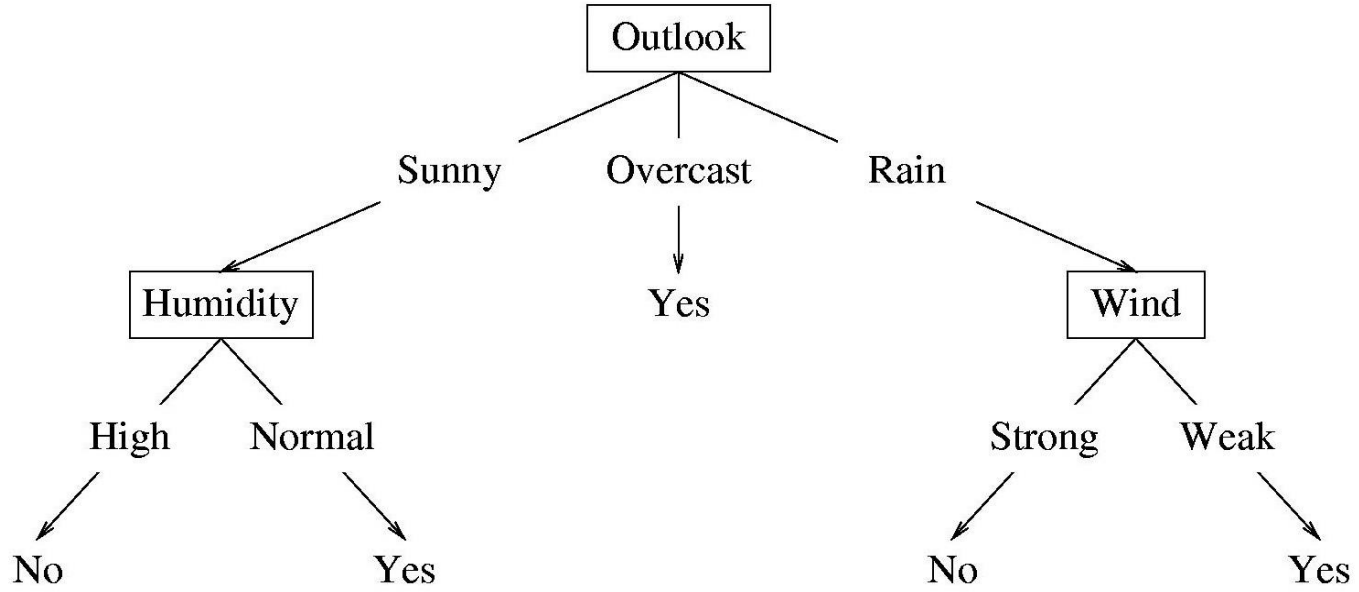
- A column denotes a feature X_i
- Rows denote labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

$\langle \boldsymbol{x}_i, y_i \rangle$

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

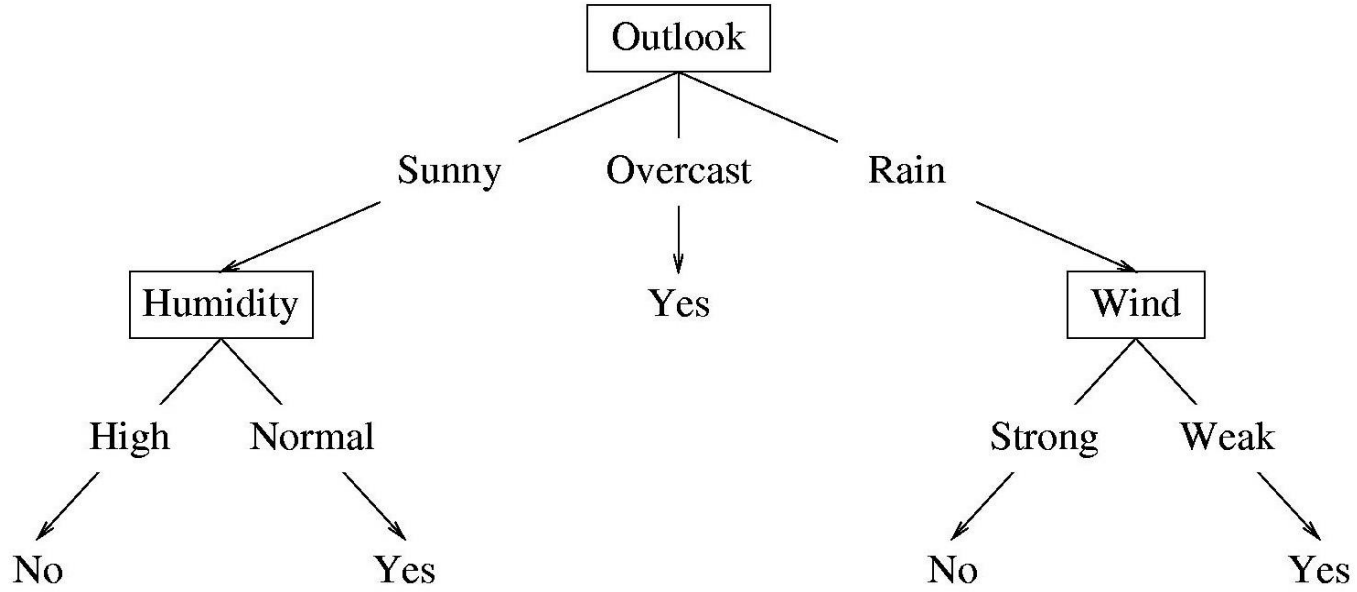
- A possible decision tree for the data:



- 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 $p(Y \mid \boldsymbol{x} \in \text{leaf})$ **)**

Decision Tree

- A possible decision tree for the data:

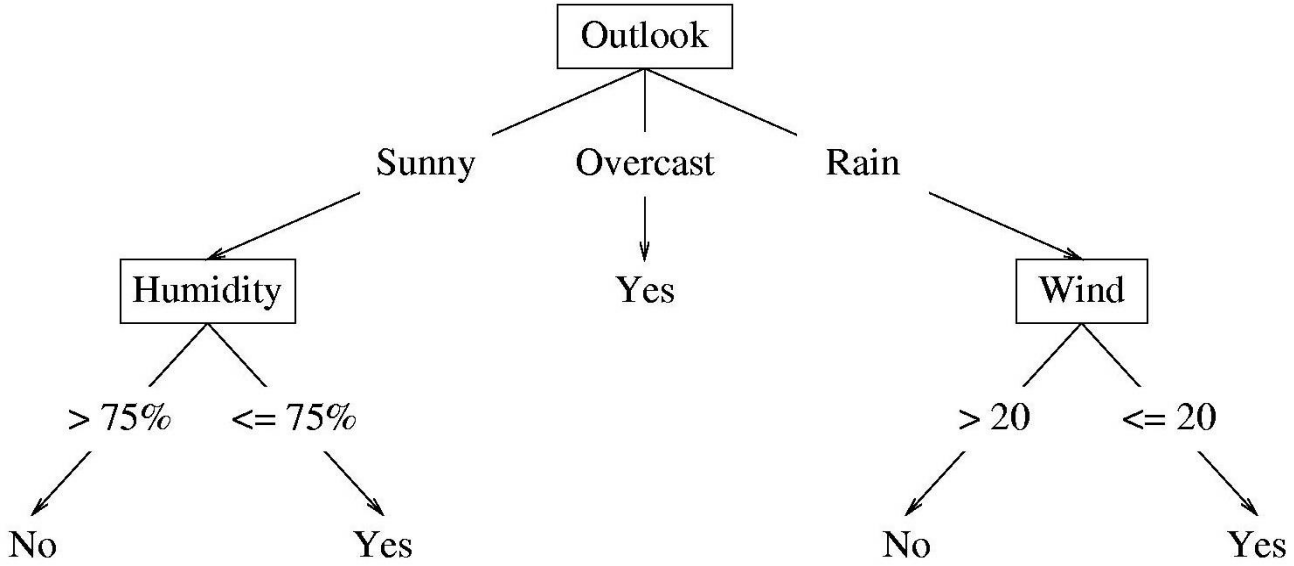


- What prediction would we make for

<outlook=sunny, temperature=hot, humidity=high, wind=weak>?

Decision Tree

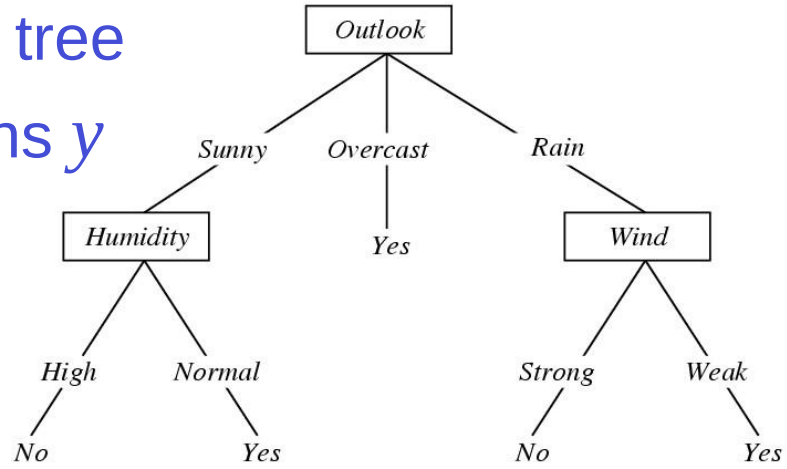
- If features are continuous, internal nodes can test the value of a feature against a threshold



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
  - tree sorts x to leaf, which assigns y



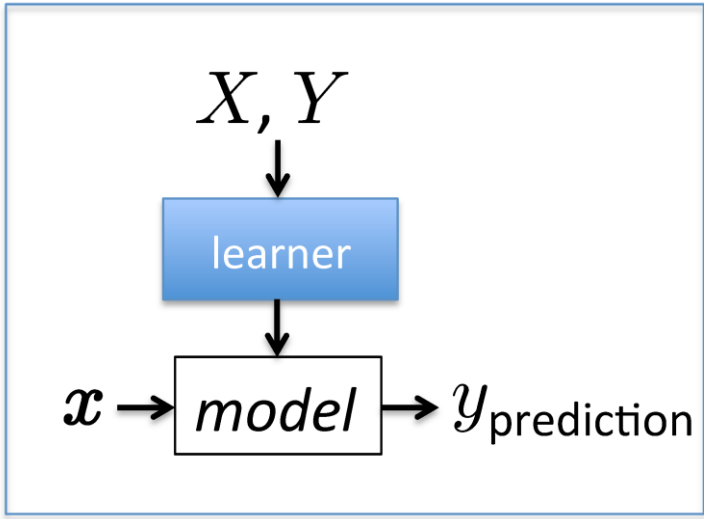
Stages of (Batch) Machine Learning

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n}$

- Assumes each $\boldsymbol{x}_i \sim \mathcal{D}(\mathcal{X})$ with $y_i = f_{target}(\boldsymbol{x}_i)$

Train the model:

model ← *classifier*.train(X, Y)



Apply the model to new data:

- Given: new unlabeled instance $\boldsymbol{x} \sim \mathcal{D}(\mathcal{X})$

$y_{\text{prediction}}$ ← *model*.predict($\boldsymbol{x}$)

Example Application: A Tree to Predict Caesarean Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

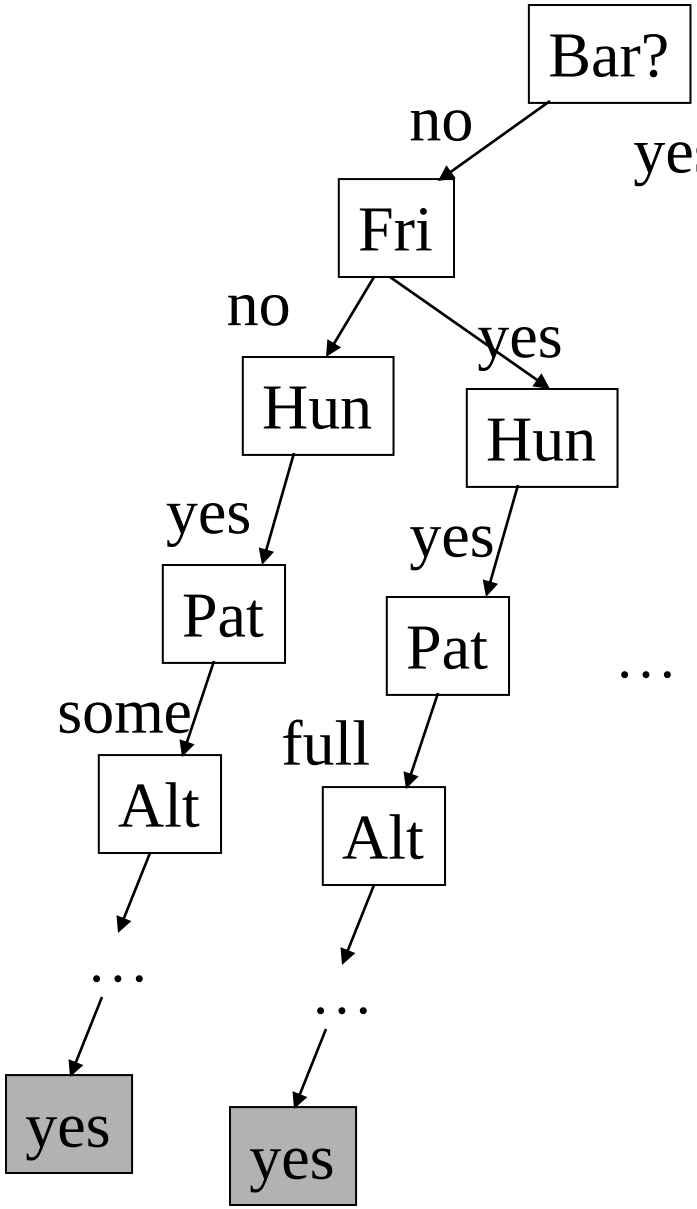
Another Example: Restaurant Domain (Russell & Norvig)

Model a patron's decision of whether to wait for a table at a restaurant

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

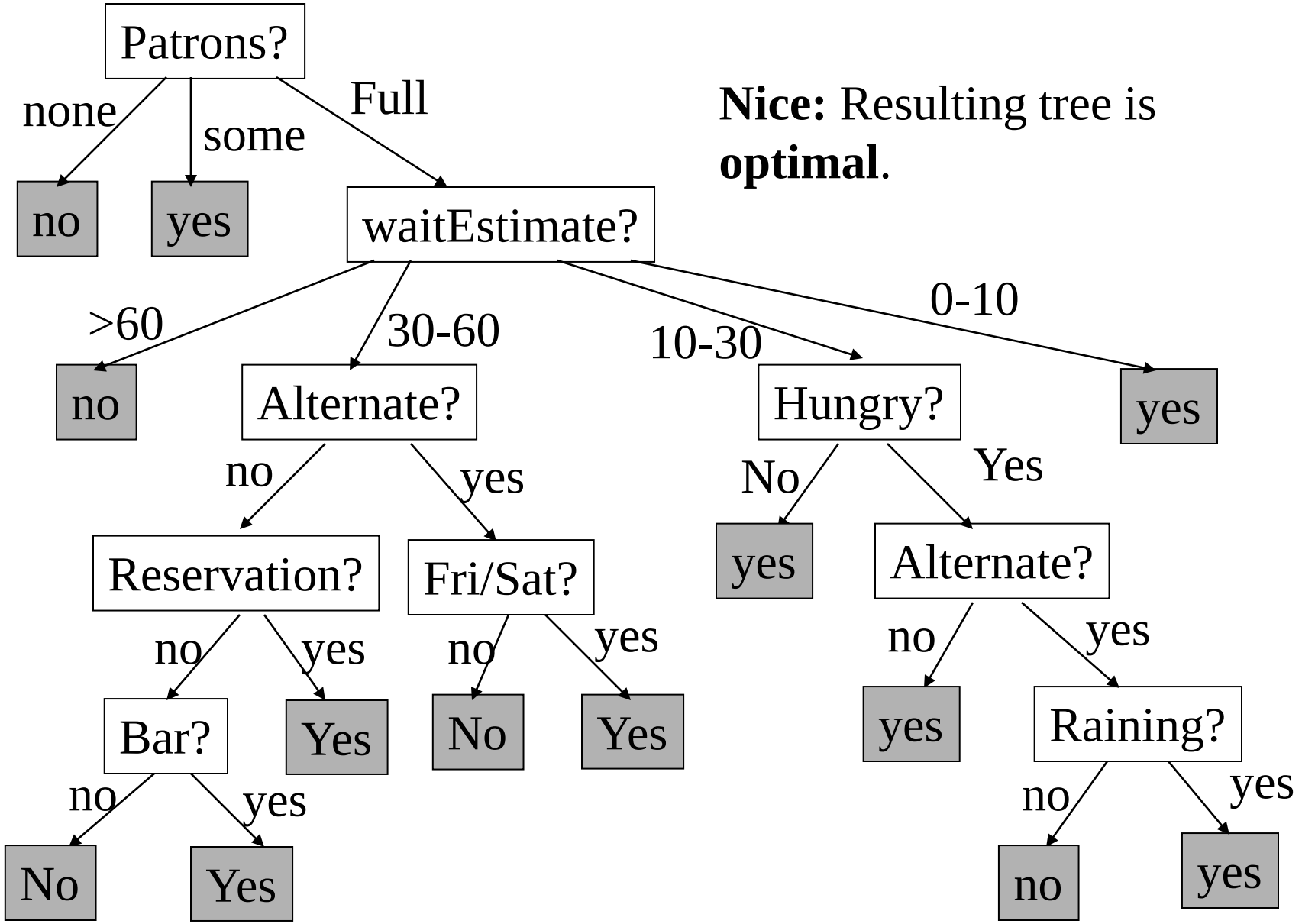
1. Alternate restaurant?
2. Bar
3. Fri/Sat?
4. Hungry
5. Patrons
6. Price range
7. Raining?
8. Reservation
9. Type of restaurant
10. Wait Estimate

Example of a Decision Tree



Problem: Resulting tree could be very long

Example of a Decision Tree (II)



Nice: Resulting tree is **optimal**.

Basic Algorithm for Top-Down Induction of Decision Trees [ID3, C4.5, …]

node = root of decision tree

Main loop:

1. *A* ← the "best" decision attribute for the next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recurse over new leaf nodes.

How do we choose which attribute is best?

Choosing the Best Attribute

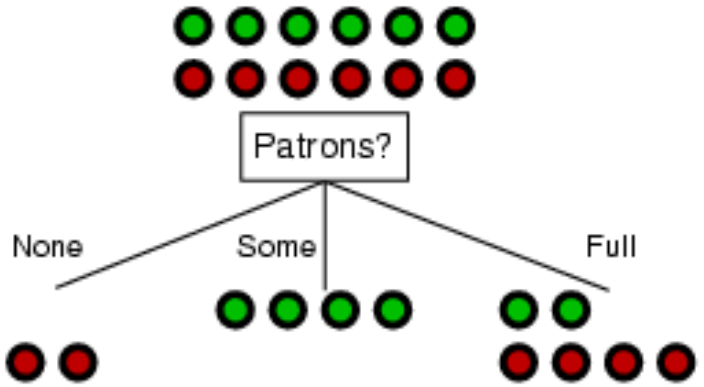
- **Key problem**: choosing which attribute to split a given set of examples

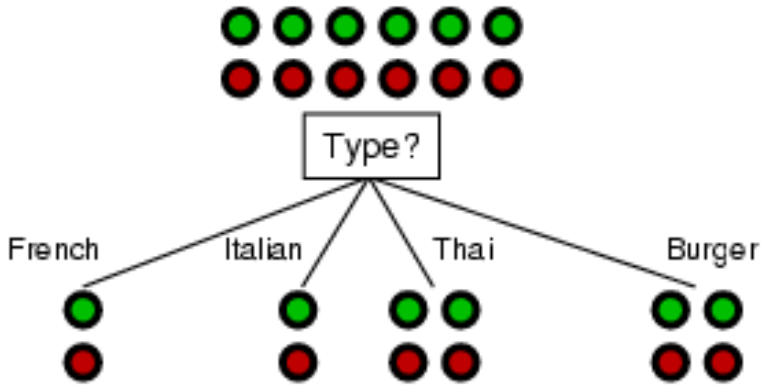
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children

- The ID3 algorithm uses the Max-Gain method of selecting the best attribute
  - ID3 is a basic algorithm for learning Decision Trees

Choosing an Attribute

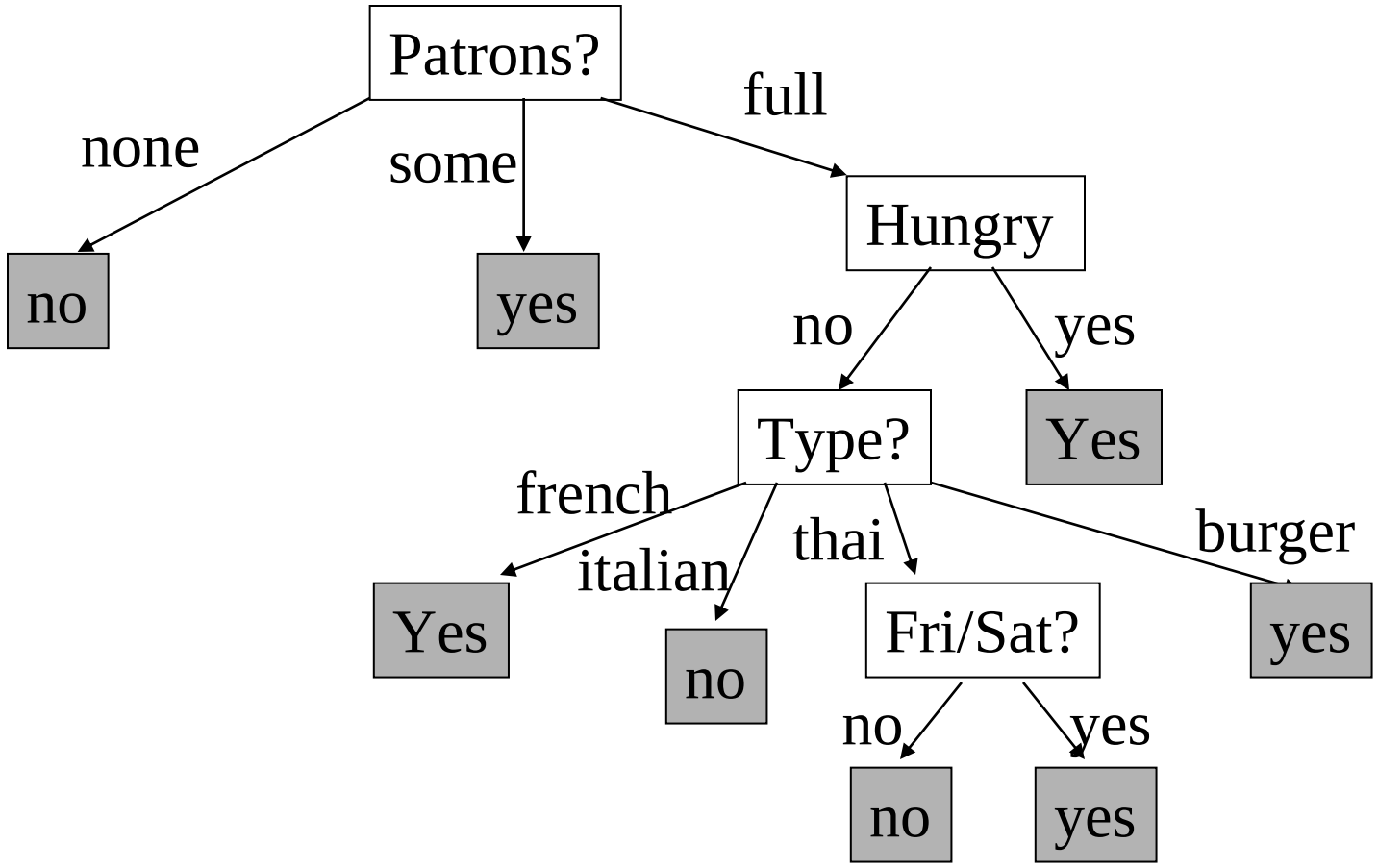
Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”



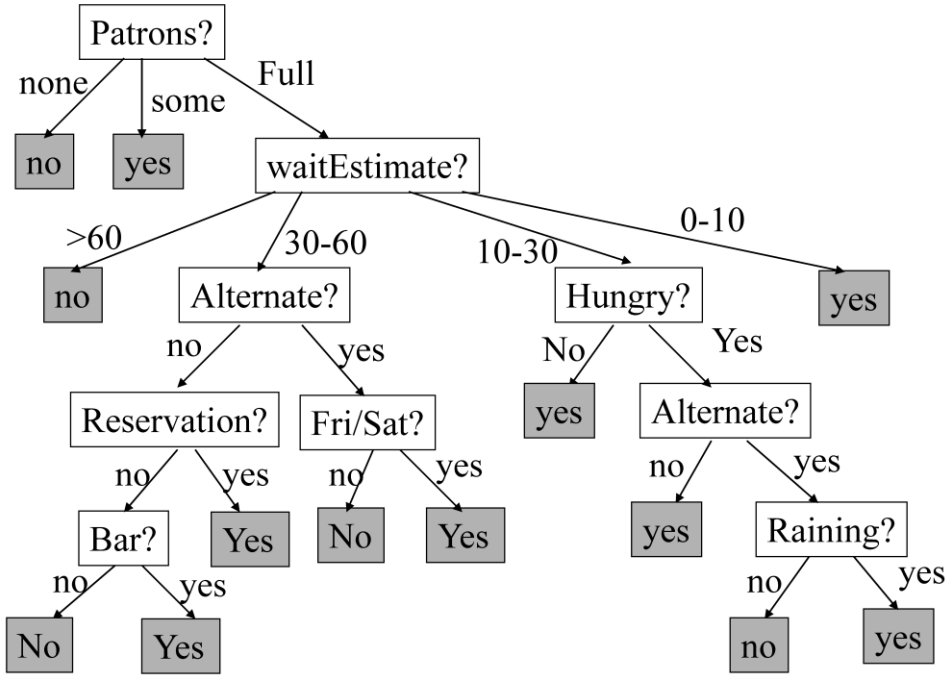


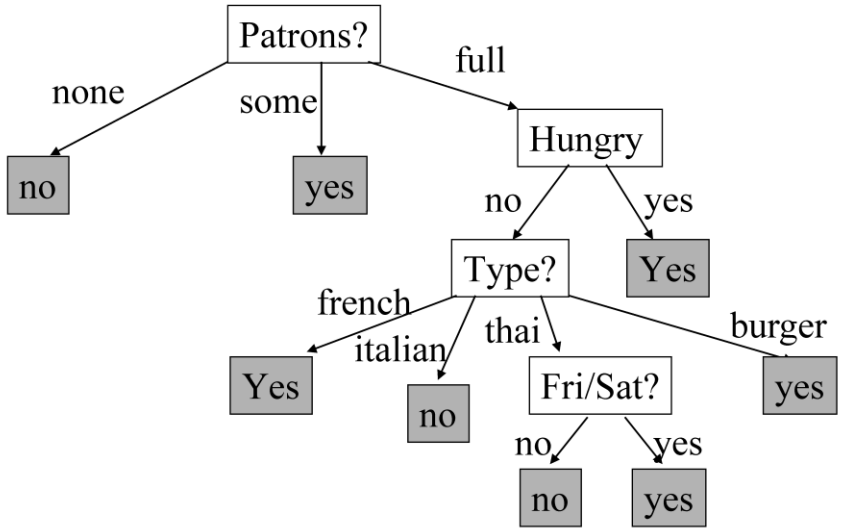
Which split is more informative: *Patrons?* or *Type?*

ID3-inducted Decision Tree

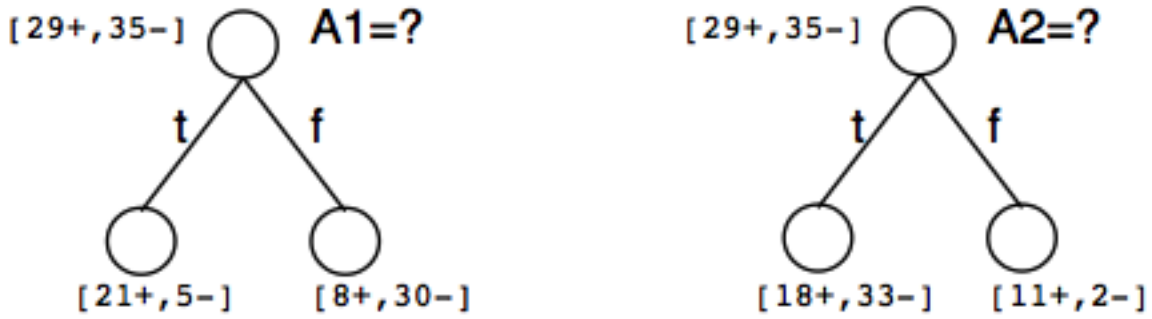


Compare the Two Decision Trees





Which attribute is the best classifier?



- A statistical property called *information gain*, measures how well a given attribute separates the training examples
- Information gain uses the notion of *entropy*, commonly used in information theory
- *Information gain = expected reduction of entropy*

Entropy

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *Y.*

$$H(Y) = -\sum_{i=1}^{n} P(Y = i) \log_2 P(Y = i)$$

- In a binary attribute,
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

$Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$ $[0 \log_2 0 = 0]$

$Entropy\ ([14+, 0-]) =$

$Entropy\ ([9+, 5-]) =$

$Entropy\ ([7+, 7-]) =$

$[\log_2 1/2 = -1]$

Entropy

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *Y.*

$$H(Y) = -\sum_{i=1}^{n} P(Y = i) \log_2 P(Y = i)$$

- In a binary attribute,
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

$$Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \qquad [0 \log_2 0 = 0]$$

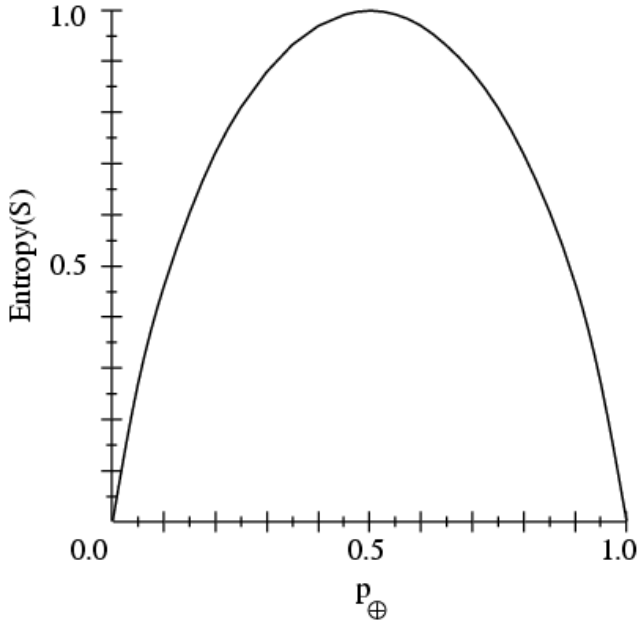
$$Entropy\ ([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy\ ([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$Entropy\ ([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) =$$

$$= 1/2 + 1/2 = 1 \qquad [\log_2 1/2 = -1]$$

Entropy



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

Conditional entropy H(Y|A)

- Given attribute A, measure Y's conditional entropy:

  = The average entropy(children nodes)

$$= \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

$Values(A)$: possible values for A

Sv: subset of S for which A has value v

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing an attribute/feature A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

$Values(A)$: possible values for A

Sv: subset of S for which A has value v

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

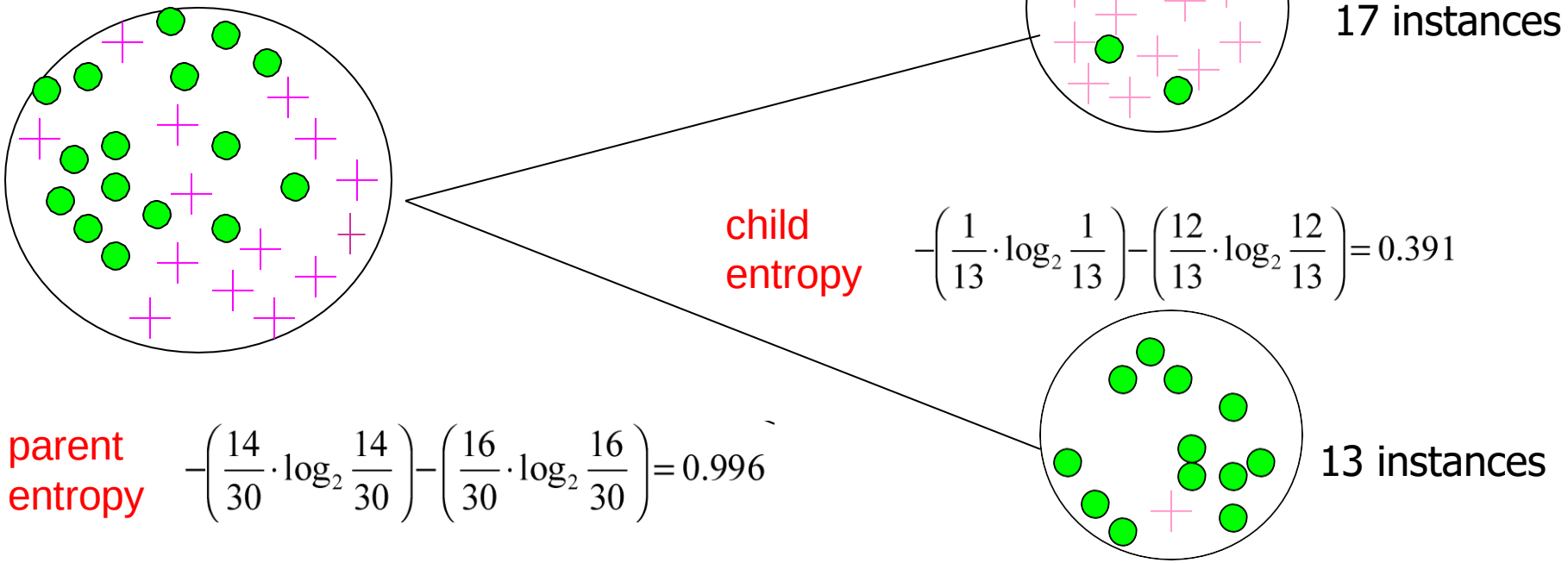
Calculating Information Gain

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

$$H(Y) = -\sum_{i=1}^{n} P(Y=i)\log_2 P(Y=i)$$

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