LEARNING

Prepared By

-Anooja Joy

LEARNING

- An agent is learning if it is improving its performance on future tasks after making observations about the world.
- Learning happens through the **course of their interactions with the world** and t**hrough the experience of their own internal states** and **processes**
- AI systems grow from a minimal amount of knowledge by learning.
- **Herbert Simon** defines learning as: ***Any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population***
- Learning depends on:
  - ***Component*** of agent to be improved.
  - ***Prior Knowledge*** that agent already has.
  - ***Representation*** used for data and component.
  - ***Feedback*** available to learn from

LEARNING ISSUES AND VIEWS

1. **Generalization from experience:**
    1. **Induction:** From a large domain, a learner usually examines a fraction of all possible examples; from this limited experience, the learner must generalize correctly to unseen instances of the domain. This is the problem of **induction.**
    2. **Inductive biases:** In most learning problems, the available data are not sufficient to guarantee optimal generalization, no matter what algorithm is used. Learners must generalize heuristically, that is, they must select those aspects of their experience that are most likely to prove effective in the future. Such selection criteria that select the most effective aspects of their experience are known as **inductive biases**
2. **Performance change in Learner:** nature of change either upgrade or degrade
    - acquisition of explicitly represented domain knowledge, based on its experience, the learner constructs or modifies expressions in a formal language (e.g. logic) and retains this knowledge for future use.

    Eg: Symbolic approaches, Neural or connectionist networks, genetic and evolutionary learning, dynamic and probabilistic

LEARNING ALGORITHMS

- Learning Algorithms vary in **goals, available training data, learning strategies** and **knowledge representation languages**.
- All algorithms learn by **searching through a space** of possible concepts to find an **acceptable generalization**.

1. **Inductive learning**

- **Learning a generalization from a set of examples**
- **concept learning** is a typical inductive learning:
  - infer a definition from given examples of some concept (e.g. cat, soybean disease)
  - allow to correctly recognize future instances of that concept
  - Two algorithms: **version space search** and **ID3**
- **Concept formation** -- clustering

LEARNING ALGORITHMS

2. Similarity-based vs. Explanation-based

Similarity-based (data-driven)

- using no prior knowledge of the domain
- rely on large numbers of examples
- generalization on the basis of patterns in training data

Explanation-based Learning(prior knowledge-driven)

- using prior knowledge of the domain to guide generalization
- learning by analogy and other technology that utilize prior knowledge to learn from a limited amount of training data

LEARNING ALGORITHMS

3. Supervised vs. Unsupervised

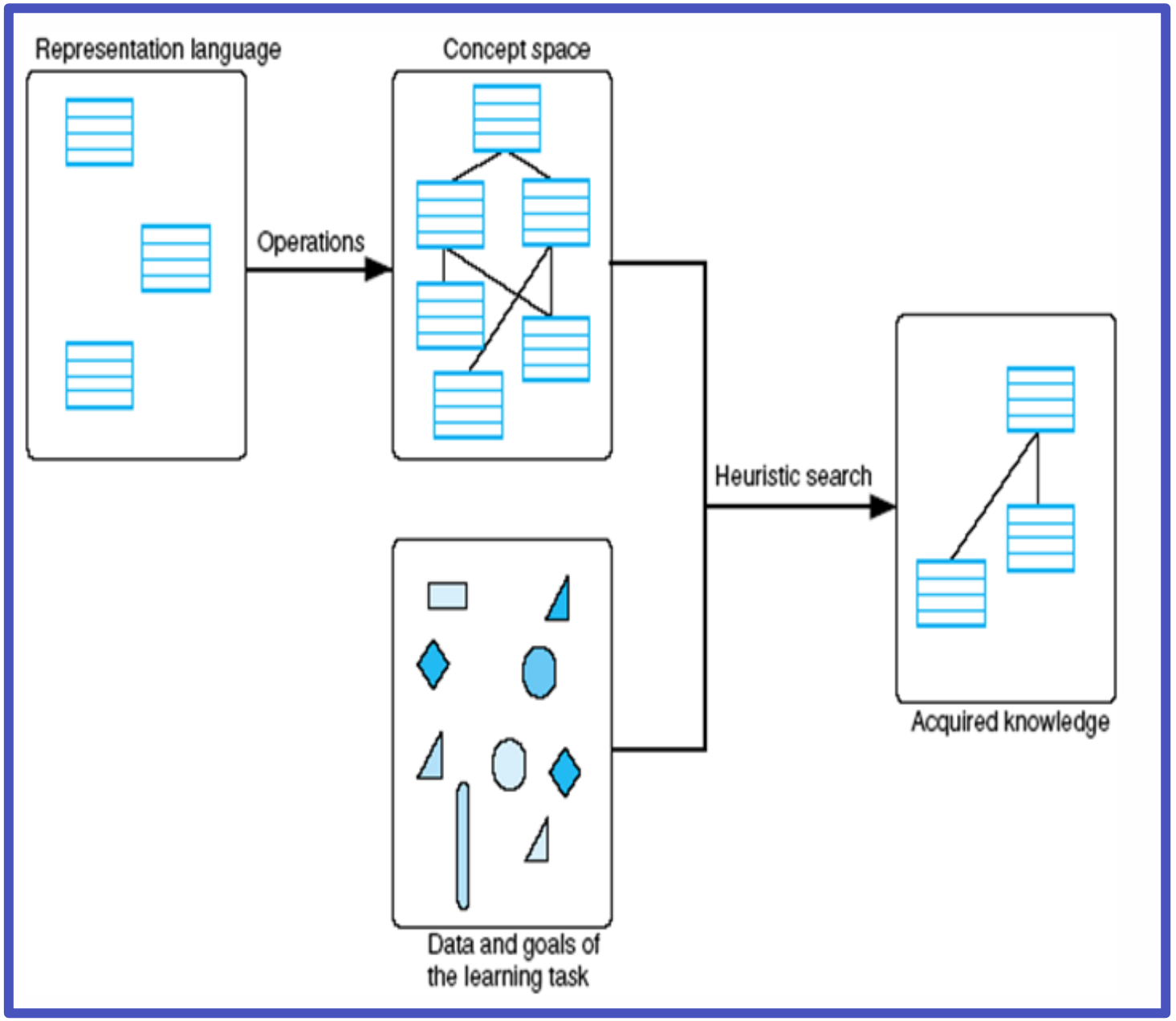
- **supervised learning**
- learning from training instances of known classification
- **unsupervised learning**
- learning from unclassified training data
- conceptual clustering or category formation

4. Reinforcement learning

A FRAMEWORK FOR SYMBOL-BASED LEARNING

- Learning algorithms may be characterized along several dimensions:

1. **Data and goals of the learning task** What are given – training instances–What are expected
2. **Representation Language**–Logic expressions – Decision trees–Rules
3. **A set of operations** Generalization/specialization – Heuristic rules –Weight adjusts
4. **Concept space** –Search space: representation, format
5. **Heuristic Search** –Search control in the concept space
6. **Acquired knowledge**



A FRAMEWORK FOR SYMBOL-BASED LEARNING

DATA AND GOALS

Type of data

positive or negative examples
Single positive example and domain specific knowledge
high-level advice (e.g. condition of loop termination)
analogies(e.g. electricity vs. water)

Goal of Learning algorithms: acquisition of

concept, general description of a class of objects
plans
problem-solving heuristics
other forms of procedural knowledge

Properties and quality of data

come from the outside environment (e.g. teacher) or generated by the program itself
reliable or contain noise
well-structured or unorganized
positive and negative or only positive

A FRAMEWORK FOR SYMBOL-BASED LEARNING

DATA AND GOALS

	Concept learning	**Explanation-based**	**Clustering**
Data	Positive/negative examples of a target class	A training example + prior knowledge	A set of unclassified instances
Goal	To infer a general definition	To infer a general concept	To discover categorizations

A FRAMEWORK FOR SYMBOL-BASED LEARNING

REPRESENTATION OF LEARNED KNOWLEDGE

concept expressions in predicate calculus

A simple formulation of the concept learning problem as conjunctive sentences containing

```
size(obj1, small) ^ color(obj1, red) ^ shape(obj1, round)
size(obj2, large) ^ color(obj2, red) ^ shape(obj2, round)
=> size(X, Y) ^ color(X, red) ^ shape(X, round)
```

structured representation such as frames

description of plans as a sequence of operations or triangle table

representation of heuristics as problem-solving rules

A FRAMEWORK FOR SYMBOL-BASED LEARNING

A SET OF OPERATIONS

- Given a set of training instances, the leaner must construct a
- generalization, heuristic rule, or plan that satisfies its goal
- Requires ability to manipulate representations
- Typical operations include
  - generalizing or specializing symbolic expressions
  - adjusting the weights in a neural network
  - modifying the program's representations

CONCEPT SPACE

- defines a space of potential concept definitions
- complexity of potential

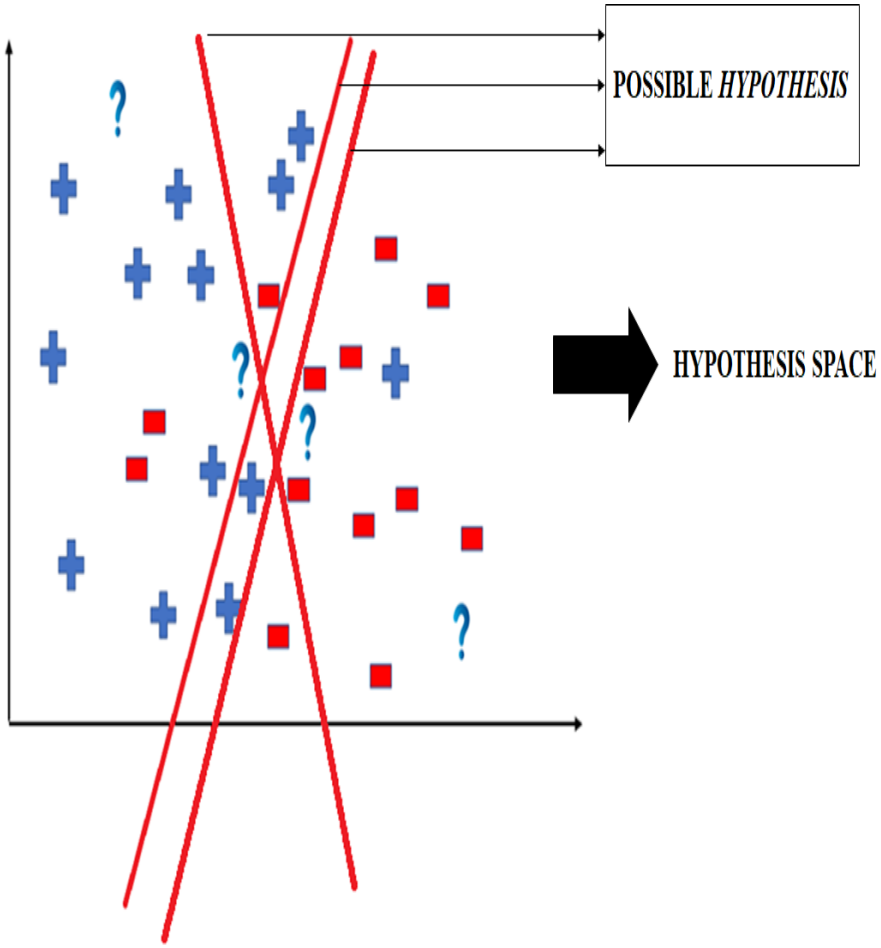
A FRAMEWORK FOR SYMBOL-BASED LEARNING

HEURISTIC SEARCH

- Use available training data and heuristics to search efficiently
- concept through generalization and specialization.
- Generalization changes the candidate description to let it accommodate new positive examples
- Specialization changes the candidate description to exclude near misses
- Performance of learning algorithm is highly sensitive to the quality and order of the training examples

UNDERSTANDING HYPOTHESIS

- In most **supervised machine learning algorithm**, our main goal is to **find out a possible hypothesis** from the **hypothesis space** that could possibly map out the inputs to the proper outputs.
- **Hypothesis** are statement about the given problem. **A hypothesis is a function that best describes the target in supervised machine learning.**
- **Hypothesis space** is the set of all the possible legal hypothesis. This is the set from which the machine learning algorithm would determine the best possible (only one) which would best describe the target function or the outputs.
- **Hypothesis testing** is a statistical method that is used in making a statistical decision using experimental data. Hypothesis testing is basically an assumption that we make about a ~~population parameter.~~



PARAMETERS OF HYPOTHESIS TESTING

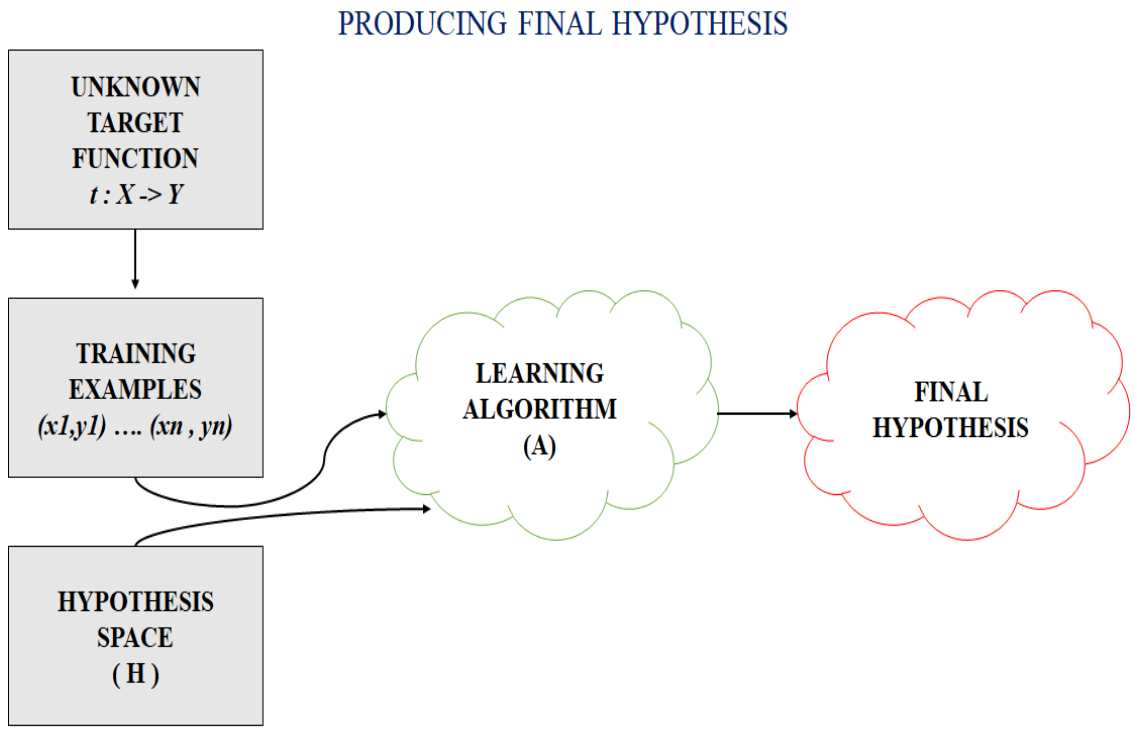
- **Null hypothesis(H0):** It is a basic assumption or made based on the problem knowledge.

Example: A company production is = 50 unit/per day etc.

- **Alternative hypothesis(H1):** The alternative hypothesis is the hypothesis used in hypothesis testing that is contrary to the null hypothesis.

Example : A company production is not equal to 50 unit/per day etc.

- **Level of significance:** It refers to the degree of significance in which we accept or reject the null-hypothesis. 100% accuracy is not possible for accepting a hypothesis, so level of significance that is usually 5% normally denoted with 0.05 or 5%, which means output should be 95% confident to give similar kind of result in each sample.



VERSION SPACE SEARCH

- A **version space** is a **hierarchical representation of knowledge that enables you to keep track of all the useful information supplied by a sequence of learning examples without remembering any of the examples.**
- **Version space** search uses inductive learning as **search through a concept space. Concept Learning** can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- The **version space**, denoted VS_H,D, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D.
- Version space search takes advantage of the fact that generalization operations impose an ordering on the concepts in a space, and then uses this ordering to guide the search.
- A **hypothesis** h is **consistent** with a set of training examples D iff **h(x) = c(x)** for each example in D.
- The **candidate-Elimination algorithm computes the version space containing all (and only those) hypotheses from H that are consistent with an observed sequence of training examples**.

VERSION SPACES AND THE CANDIDATE ELIMINATION ALGORITHM

- **Instead of enumerating all the hypotheses consistent with a training set,** we can represent its most specific and most general boundaries. The hypotheses included in-between these two boundaries can be generated as needed.
- The **general boundary G,** with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D. It is also known as **General Hypothesis** which is **not specifying features to learn the machine.** **G = {'?', '?','?','?'...}:** Number of attributes
- The **specific boundary S,** with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D. It is also known as **Specific Hypothesis** which is specifying features to learn machine (Specific feature) **S= {'pi','pi','pi'...}:** Number of pi depends on number of attributes.

CANDIDATE ELIMINATION ALGORITHM

- **Candidate elimination algorithm** incrementally builds the version space given a hypothesis space ***H*** and a set *E* of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.
- **Most general hypothesis:** (?, ?, ?, ?, ?) . The **General boundary, G,** of version space V SH,D is the set of its maximally general members that are consistent with the given training set. G is maximally general hypotheses in H
- **Most specific hypothesis:** (∅, ∅, ∅, ∅, ∅). The **Specific boundary, S,** of version space V SH,D is the set of its maximally specific members that are consistent with the given training set . S is maximally specific hypotheses in H
- **Hypotheses H** is a Conjunctions of literals. E.g. < ?, Cold, High, ?, ?, ? >.**Every member of the version space lies between these boundaries**
- Candidate elimination algorithm helps in determining a hypothesis h in H with h (x) = c (x) [consistent] for all x in Training Set D.

Step1: Initialize General Hypothesis to contain one element: the null description (all features are variables). [[?, ?, ?, ?, ?, ?]]
Step2: Initialize Specific Hypothesis, to contain one element: the first positive example.
Step3: For each training example
Step4: If example is **positive example**

1.Generalize all the **specific models** to match the positive example and by **replacing attribute value with '? For a controversy,** but ensure the following:

- The new specific models involve minimal changes.
- Each new specific model is a specialization of some general model.
- No new specific model is a generalization of some other specific model.

2. **Prune away** all the **general models that fail to match the positive example**.

Step5: If example is **Negative example**

- 1. Specialize all **general models to prevent match with the negative example**, by enumerating but ensure the following:
  - The new general models involve minimal changes.
  - Each new general model is a generalization of some specific model.
  - No new general model is a specialization of some other general model.

2. Prune away all the specific models that match the negative example.

Step6: If **S** and **G** are both **singleton sets**, then:

- if they are identical, output their value and halt.
- if they are different, the training cases were inconsistent. Output this result and halt.
- else continue accepting new training examples.

EXAMPLE

- Each constraint can be
  - a specific value (e.g., Water = Warm)
  - don't care (e.g., "Water =?")
  - no value allowed (e.g.,"Water= ∅")
- Examples: "Days at which my friend Aldo enjoys his favorite water sport"
- Result: classifier for days = description of Aldo's behavior

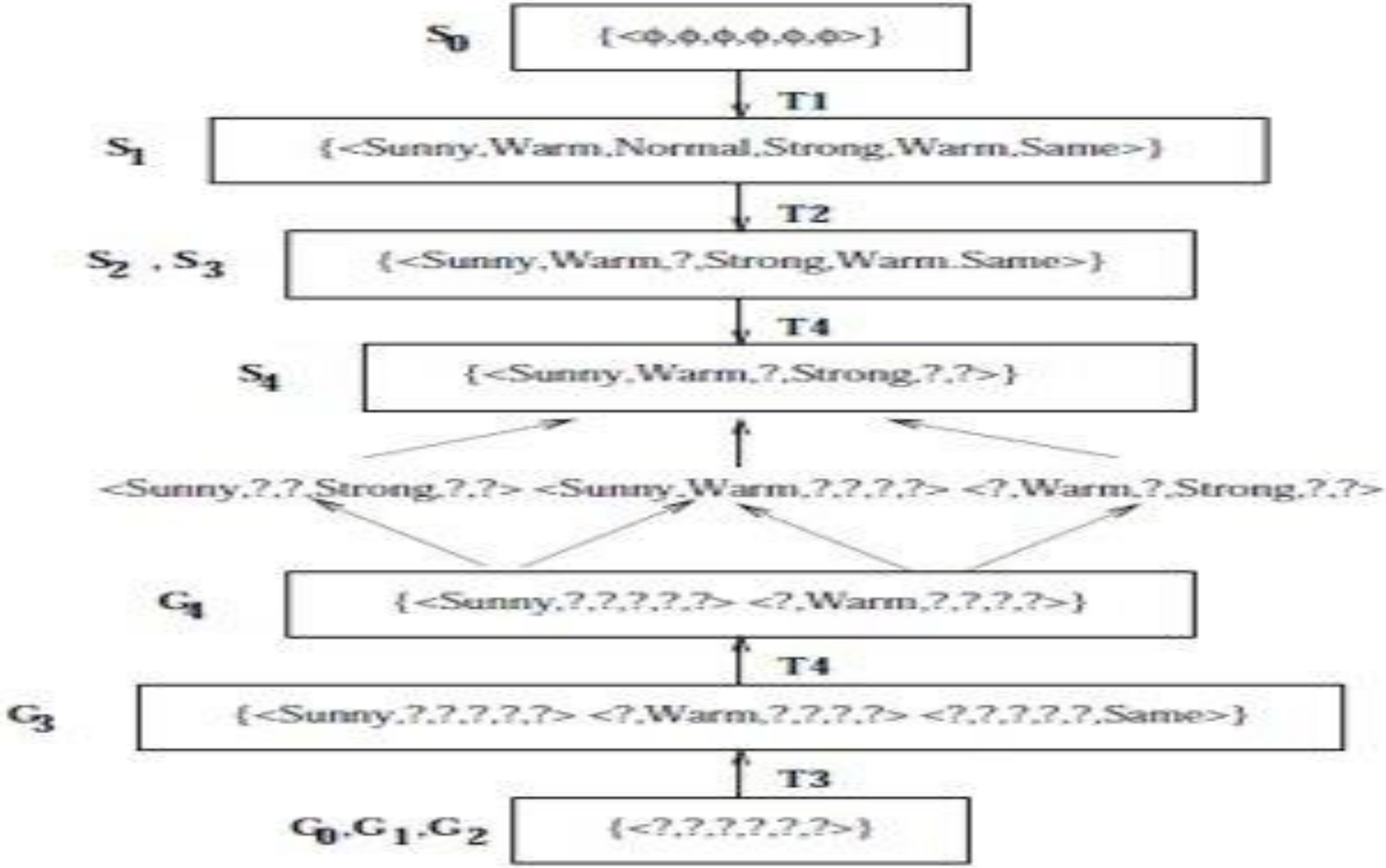
Train

	Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
T1:	*Sunny*	*Warm*	*Normal*	*Strong*	*Warm*	*Same*	*Yes*
T2:	*Sunny*	*Warm*	*High*	*Strong*	*Warm*	*Same*	*Yes*
T3:	*Rainy*	*Cold*	*High*	*Strong*	*Warm*	*Change*	*No*
T4:	*Sunny*	*Warm*	*High*	*Strong*	*Cool*	*Change*	*Yes*

THE CANDIDATE-ELIMINATION ALGORITHM: EXAMPLE

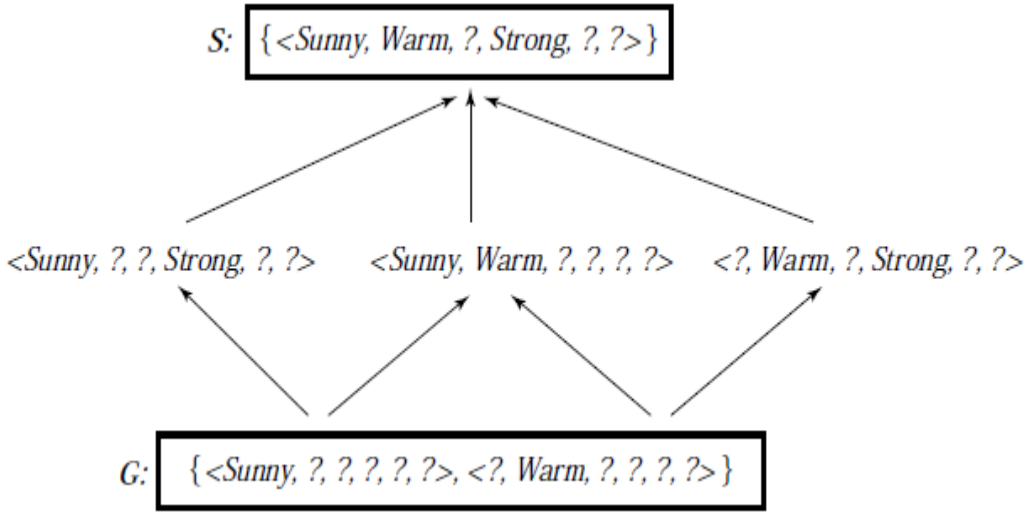
Initially : Initialize G to a singleton set that includes everything. Initialize S to null.
G = [[?, ?, ?, ?, ?, ?]]
S0 = [Null, Null, Null, Null, Null, Null]
For instance 1 : <'sunny','warm','normal','strong','warm ','same'> and positive output.
G1 = G
S1 = ['sunny','warm','normal','strong','warm ','same'] S-boundary is very specific and fails to cover positive example, hence must be generalized. Initialize S to a singleton set that includes the first positive example.
For instance 2 : <'sunny','warm','high','strong','warm ','same'> and positive output.
G2 = G
S2 = ['sunny','warm',?,'strong','warm ','same'] Make changes to S. Attribute value at 3 not same so replaced with ?.
For instance 3 : <'rainy','cold','high','strong','warm ','change'> and negative output. G boundary is overly generalised hence specialise.
G3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, 'same']]
S3 = S2 ['sunny','warm',?,'strong','warm ','same']
For instance 4 : <'sunny','warm','high','strong','cool','change'> and positive output.
Prune G to exclude descriptions inconsistent with the positive example.
[['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?],
S4 = ['sunny','warm',?,'strong', ?, ?]
At last, by synchronizing the G4 and S4 algorithm produce the output.

Output :
G = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?]]
S = ['sunny','warm',?,'strong', ?, ?]


S_0
{<ϕ,ϕ,ϕ,ϕ,ϕ,ϕ>}
T1
S_1
{<Sunny,Warm,Normal,Strong,Warm,Same>}
T2
S_2 , S_3
{<Sunny,Warm,?,Strong,Warm,Same>}
T4
S_4
{<Sunny,Warm,?,Strong,?,?>}
<Sunny,?,?,Strong,?,?> <Sunny,Warm,?,?,?,?> <?,Warm,?,Strong,?,?>
G_4
{<Sunny,?,?,?,?,?> <?,Warm,?,?,?,?>}
T4
G_3
{<Sunny,?,?,?,?,?> <?,Warm,?,?,?,?> <?,?,?,?,?,Same>}
T3
G_0,G_1,G_2
{<?,?,?,?,?,?>}

RESULTING VERSION SPACE

- The fourth training example further generalizes the S boundary of the version space.
- It also results in removing one member of the G boundary, because this member fails to cover the new positive example
- the boundary sets S4 and G4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples
- This learned version space is independent of the sequence in which the training examples are presented (because in the end it contains all hypotheses consistent with the set of examples).
- As further training data is encountered, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.



CHALLENGE

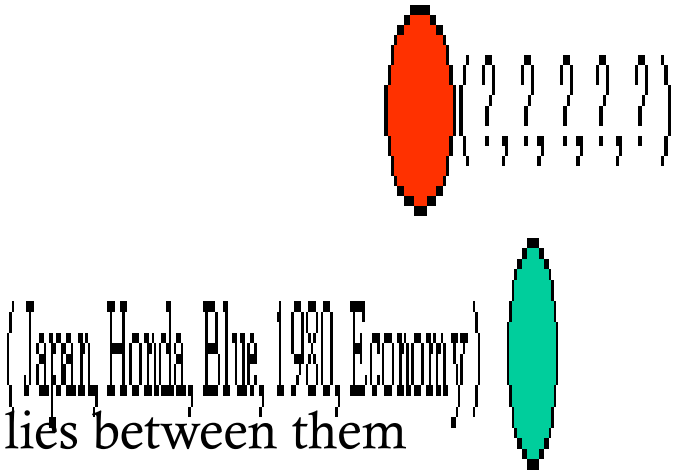
- Learning the concept of "Japanese Economy of Car"
- **Features:** Country of Origin, Manufacturer,Color, Decade, Type

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

EXAMPLE

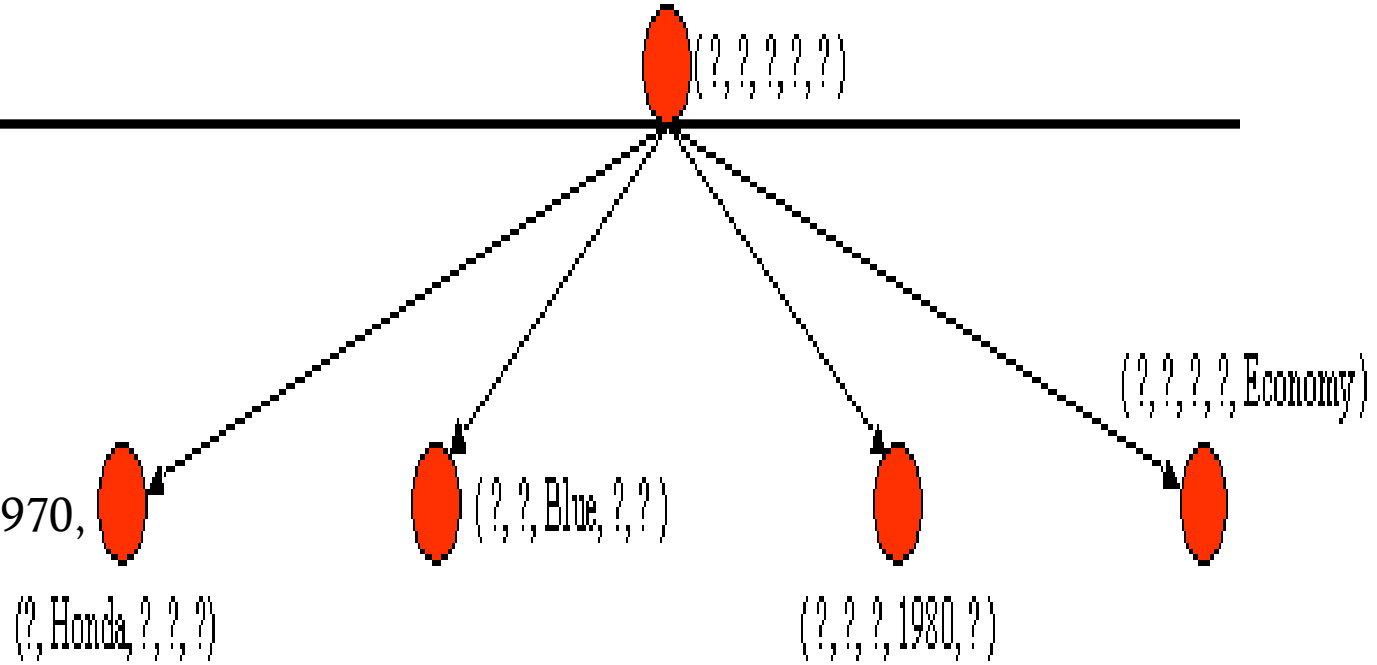
- STEP 1 **Positive Example**: (Japan, Honda, Blue, 1980, Economy)

Initialize G to a singleton set that includes everything. **Initialize S to a singleton set that includes the first positive example.**	**G = { (?, ?, ?, ?, ?) }** **S = { (Japan, Honda, Blue, 1980, Economy) }**



- The actual heuristic to be learned, "Japanese Economy Car", probably lies between them somewhere within the version space.

EXAMPLE



STEP 2 Negative Example: (Japan, Toyota, Green, 1970, Sports)

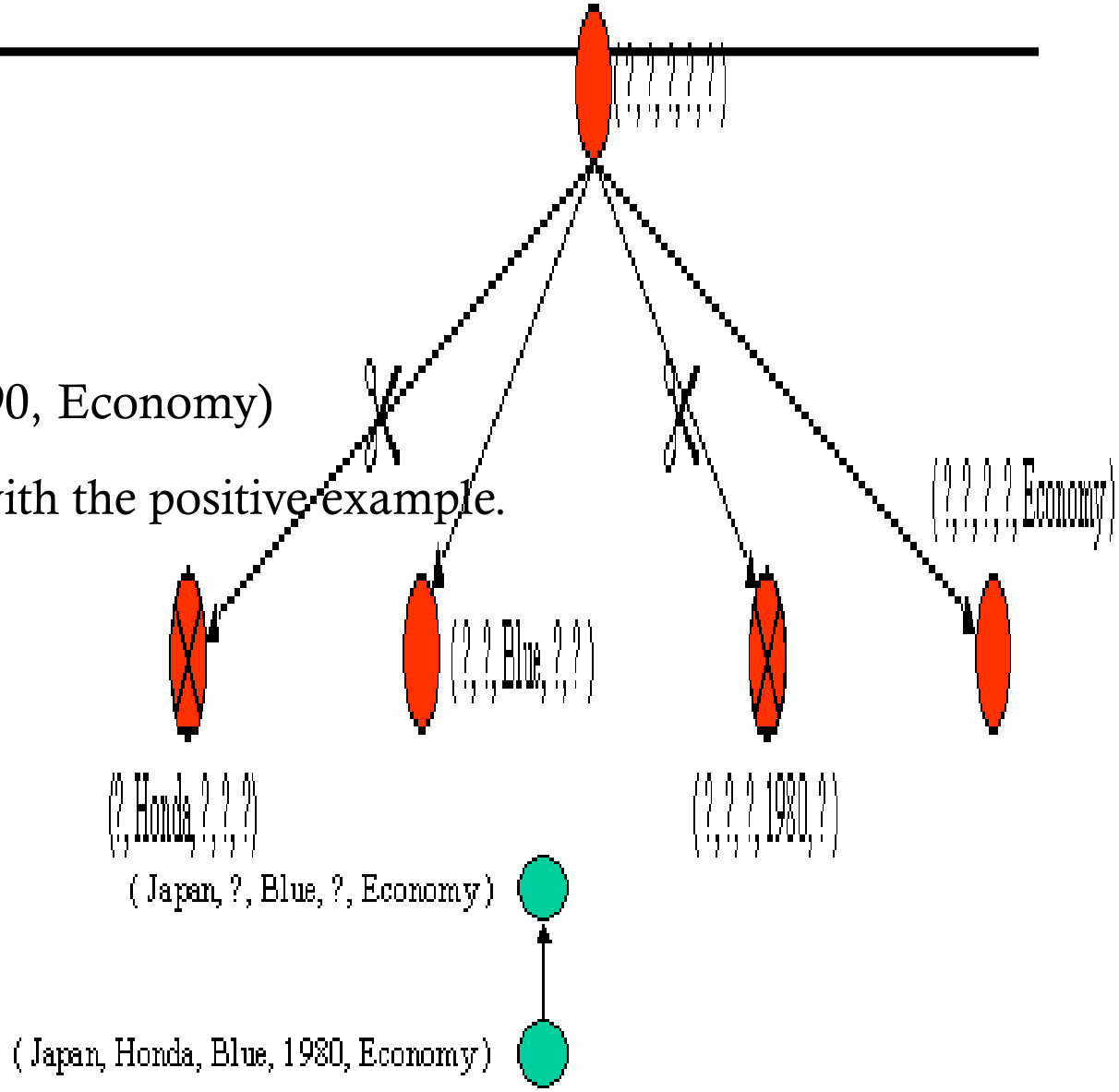
G =	**{ (?, Honda, ?, ?, ?), (?, ?, Blue, ?, ?), (?, ?, ?, 1980, ?), (?, ?, ?, ?, Economy) }**
S =	{ (Japan, Honda, Blue, 1980, Economy) }

Specialize G to exclude the negative example.

EXAMPLE

- 3. **Positive Example**: (Japan, Toyota, Blue, 1990, Economy)
- Prune G to exclude descriptions inconsistent with the positive example.
- Generalize S to include the positive example.

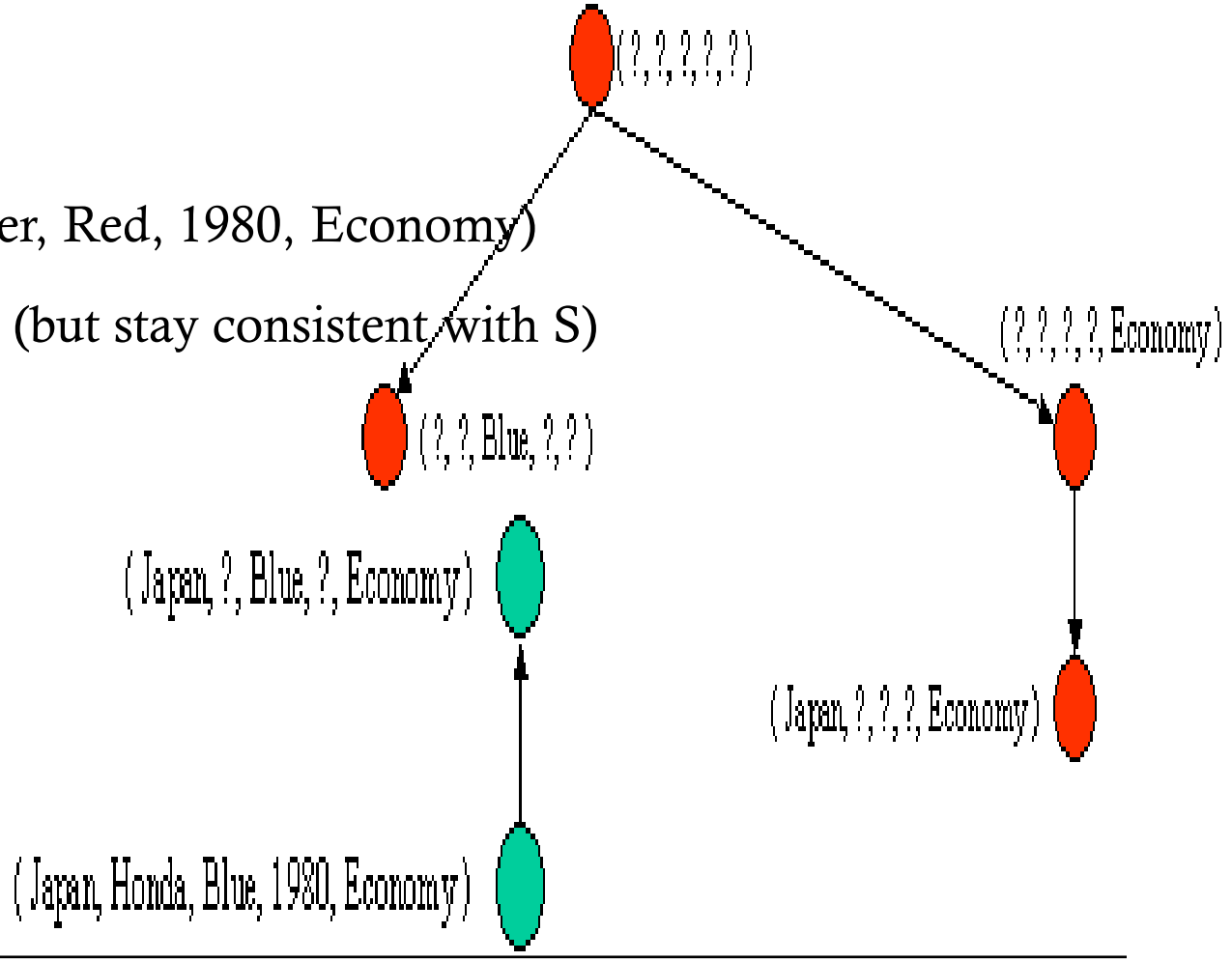
G =	**{ (?, ?, Blue, ?, ?), (?, ?, ?, ?, Economy) }**
S =	{ (Japan, ?, Blue, ?, Economy) }



EXAMPLE

- STEPN 4. **Negative Example**: (USA, Chrysler, Red, 1980, Economy)
- Specialize G to exclude the negative example (but stay consistent with S)

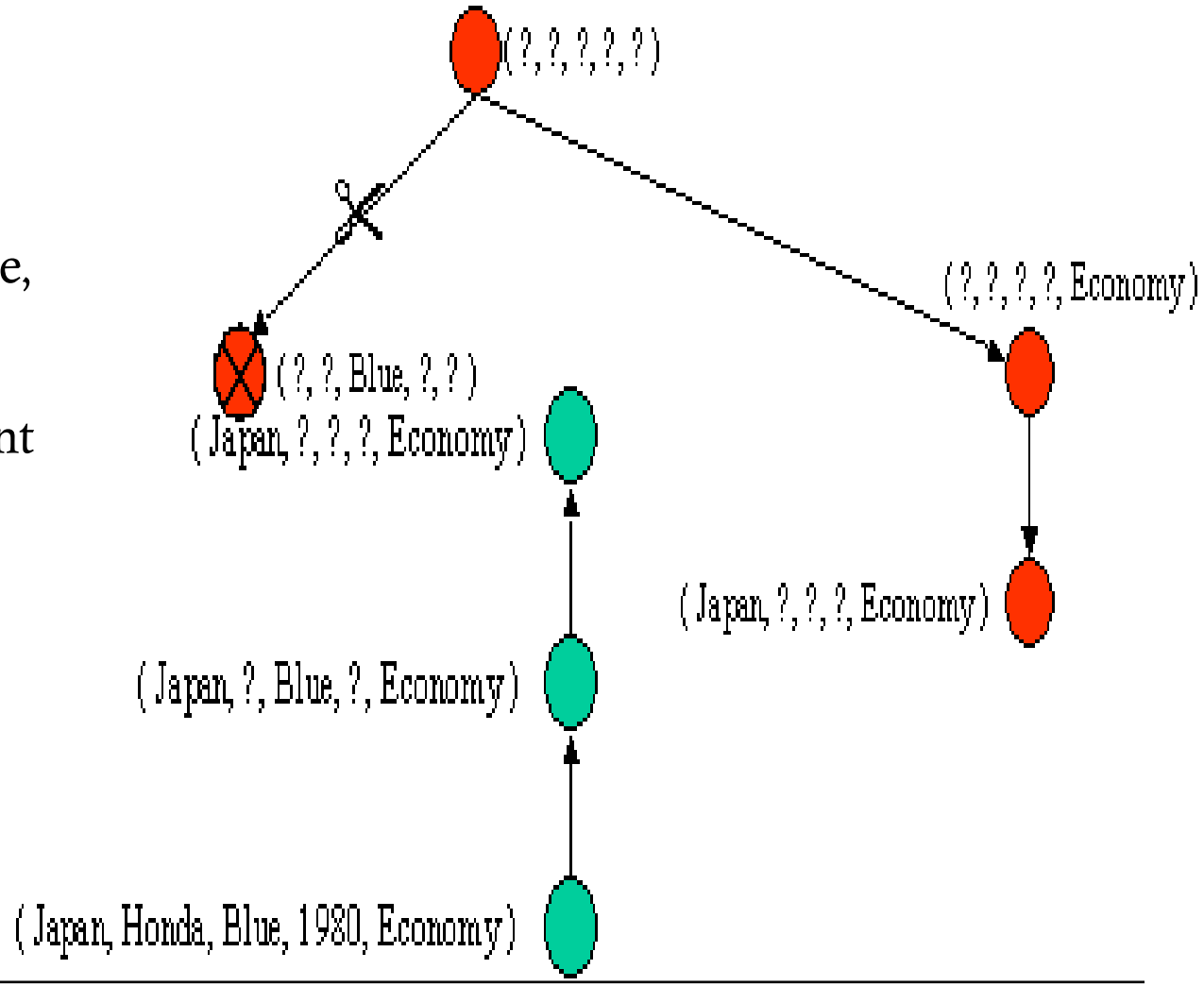
G =	**{ (?, ?, Blue, ?, ?), (Japan, ?, ?, ?, Economy) }**
S =	{ (Japan, ?, Blue, ?, Economy) }



EXAMPLE

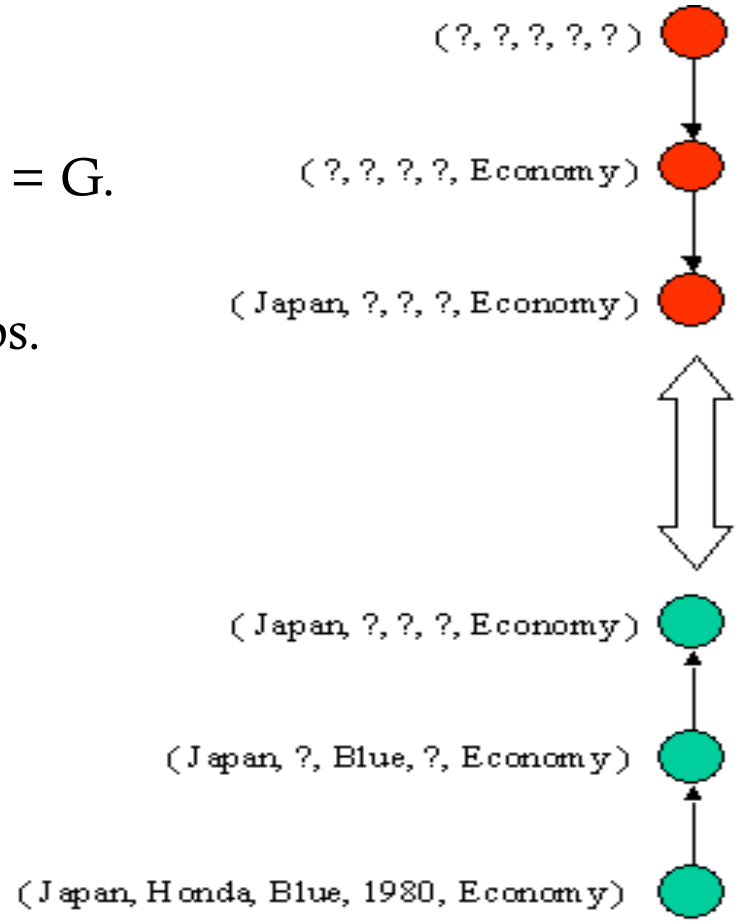
- 5. **Positive Example**: (Japan, Honda, White, 1980, Economy)
- Prune G to exclude descriptions inconsistent with positive example.
  Generalize S to include positive example.

G = { (Japan, ?, ?, ?, Economy) }
S = { (Japan, ?, ?, ?, Economy) }



EXAMPLE

- G and S are singleton sets and S = G.
  Converged.
  No more data, so algorithm stops.



CHALLENGE

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

VERSION SPACE

- **Generalization and Specialization Leads to Version Space Convergence**
- The key idea in version space learning is that specialization of the general models and generalization of the specific models may ultimately lead to just one correct model that matches all observed positive examples and does not match any negative examples.
- That is, each time a negative example is used to specialilize the general models, those specific models that match the negative example are eliminated and each time a positive example is used to generalize the specific models, those general models that fail to match the positive example are eliminated. Eventually, the positive and negative examples may be such that only one general model and one identical specific model survive.
- version space learned by Candidate-Elimination algorithm will converge towards correct hypothesis provided:
  - no errors in training examples
  - there is a hypothesis in H that describes target concept
- What if no concept in H that describes the target concept?

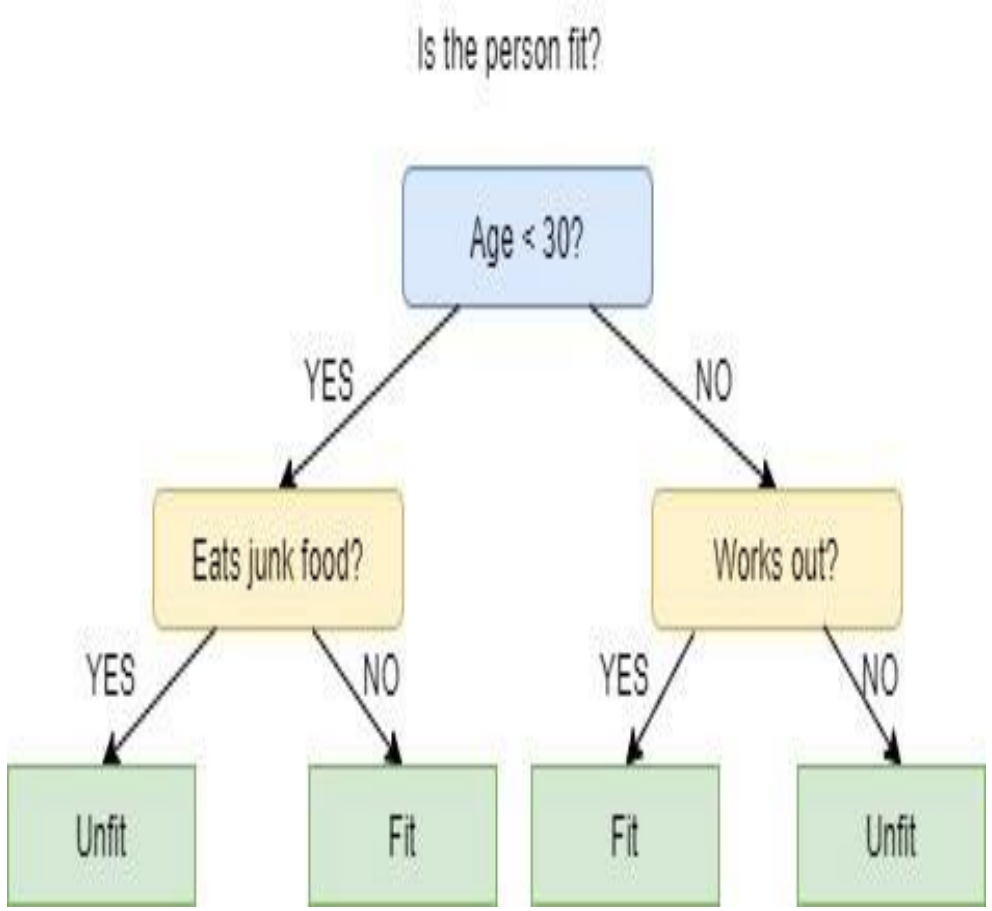

DECISION TREES

THE ID3 DECISION TREE INDUCTION ALGORITHM

- **ID3 induces concepts from examples.** This approach known as **supervised** and **non-parametric decision tree type.**
- Mostly, it is used for **classification** and **regression**.
- **ID3 represents concepts as decision trees.**
- **Decision tree:** a representation that allows us to determine the classification of an object by testing its values for certain properties
- A decision tree is a structure that includes a **root node, internal nodes, branches,** and **leaf nodes**. Each **internal node** denotes **a test on an attribute**, each **branch** denotes the **outcome of a test ie possible value ,** and each **leaf node** holds a **classification which does label**. The topmost node in the tree is the root node.
- **AIM: An individual of unknown type may be classified by traversing the decision tree.**

DECISION TREE

- Adecision tree that is used to classify whether a person is **Fit** or **Unfit.**
- **The decision nodes here are questions like ‘’** *‘Is the person less than 30 years of age?’*, *‘Does the person eat junk?’*, etc. And the leaves are one of the two possible outcomes viz. **Fit** and **Unfit**.



Data Description

The sample data used by ID3 has certain requirements, which are:

- **Attribute-value description** - the same attributes must describe each example and have a fixed number of values.
- **Predefined classes** - an example's attributes must already be defined, that is, they are not learned by ID3.
- **Discrete classes** - classes must be sharply delineated. Continuous classes broken up into vague categories such as a metal being "hard, quite hard, flexible, soft, quite soft" are not expected.
- **Sufficient examples** - since inductive generalization is used (i.e. not provable) there must be enough test cases to distinguish valid patterns from chance occurrences.

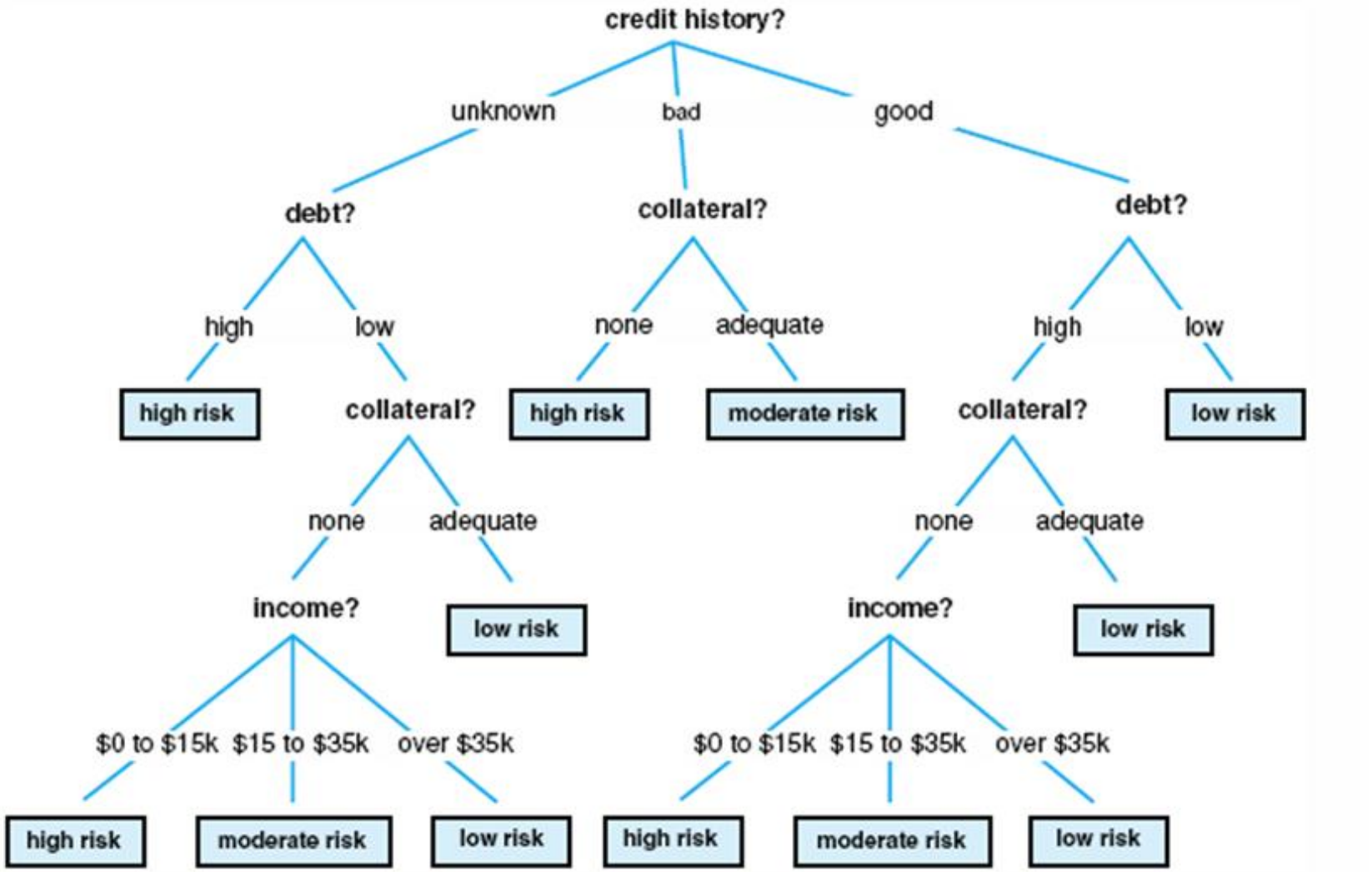
DATA FROM CREDIT HISTORY OF LOAN APPLICATIONS

Example: problem of estimating an individual's credit risk on the basis of credit history, current debt, collateral, and income(TAble 9.1)

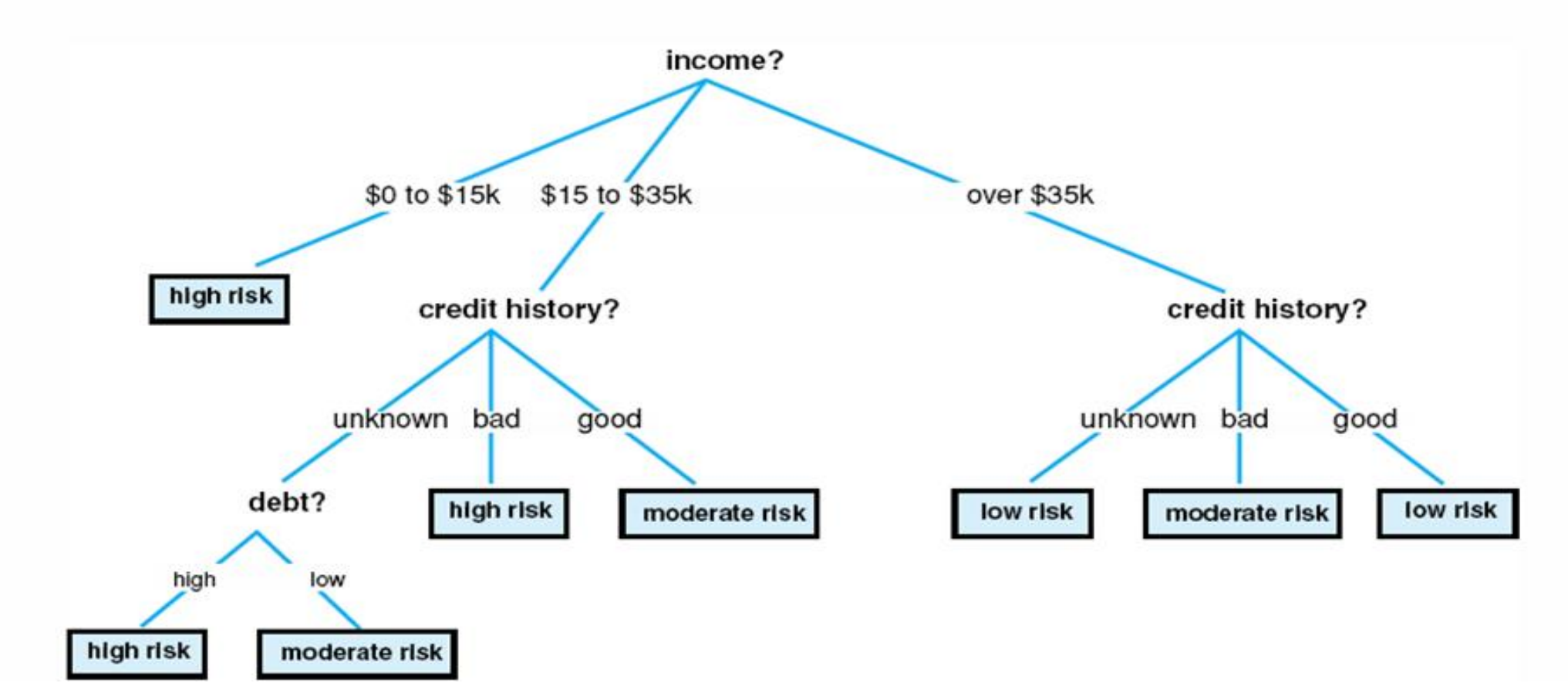
NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

DECISION TREE

- **Internal node-** ***collateral*** **or** ***debt*****or** ***income***
- **Branch-** *high* or *low*
- **Leaf nodes-** *low* or *moderate risk*



SIMPLIFIED DECISION TREE



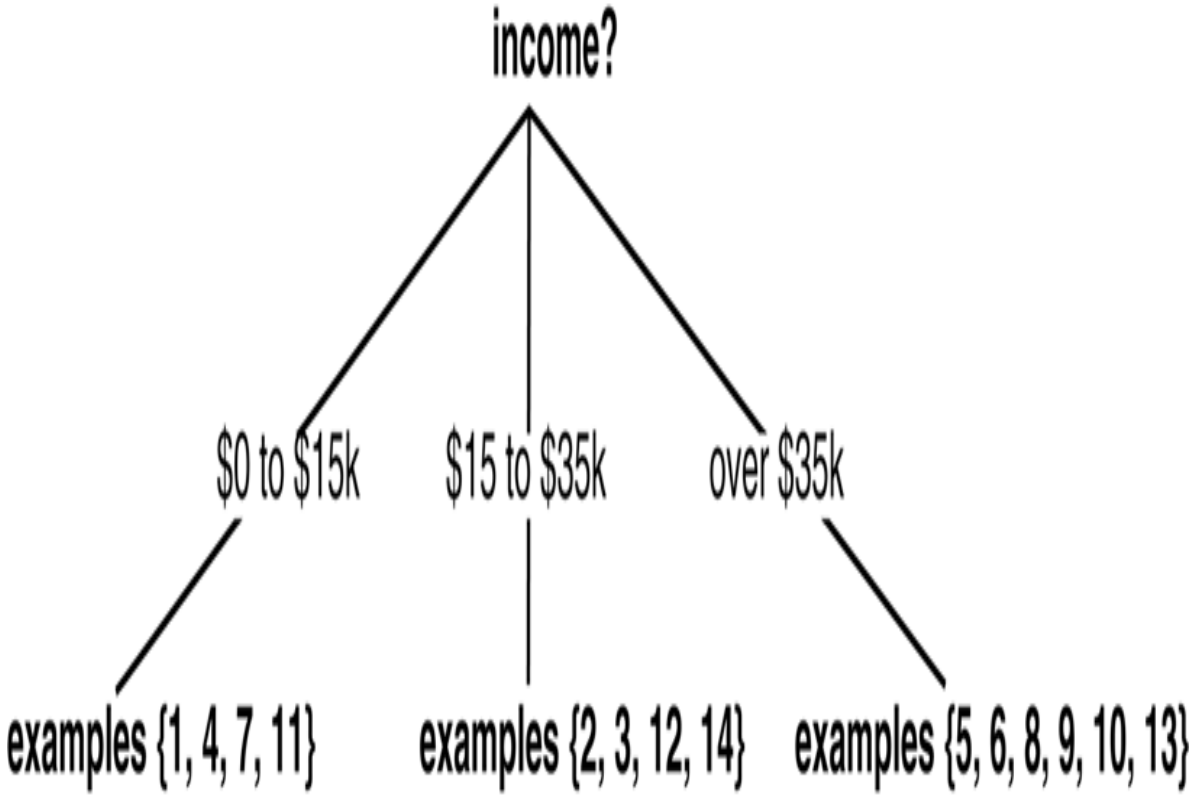
CHOICE OF THE OPTIMAL TREE

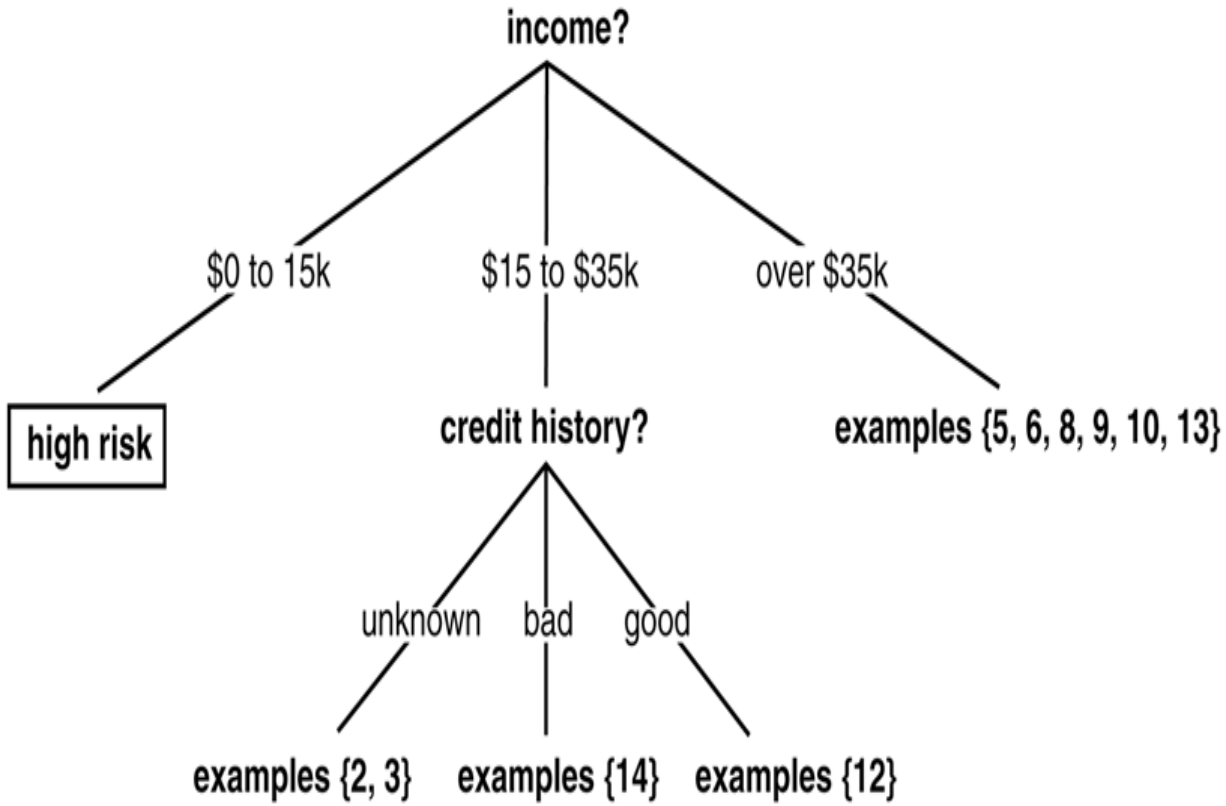
- **The size of the tree necessary to classify a given set of examples varies according to the order with which properties are tested.**
- **Measurement: the greatest likelihood of correctly classifying unseen data.**
- assumption of ID3 algorithm: **"the simplest decision tree that covers all the training examples" is the optimal tree**
- **rationale for this assumption is time-honored heuristic of preferring simplicity & avoiding unnecessary assumptions**

> *" It is vain to do with more what can be done with less…. Entities should not be multiplied beyond necessity "* Occam's Razor principle

TOP-DOWN DECISION TREE INDUCTION ALGORITHM

1. constructs decision tree in a top-down fashion
2. selects a property at the current node of the tree
3. using the property to partition the set of examples
4. recursively construct a subtree for each partition
5. Continues until all members of the partition are in the same class
6. Because the order of tests is critical, ID3 relies on its criteria for selecting the test


income?
$0 to $15k
$15 to $35k
over $35k
examples {1, 4, 7, 11}
examples {2, 3, 12, 14}
examples {5, 6, 8, 9, 10, 13}


income?
$0 to 15k
$15 to $35k
over $35k
high risk
credit history?
examples {5, 6, 8, 9, 10, 13}
unknown
bad
good
examples {2, 3}
examples {14}
examples {12}

EXAMPLE

TEST SELECTION STRATEGY

- **How does ID3 decide which attribute is the best?**
- A statistical property, called **information gain,** is used. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. Information gain for a property P ***Gain(P)= I(C)-E(P),*** where **I(C)**=Total information content of tree **E(P)**=expected information to complete tree.
- Assume a **set of training instances**, **C**. If we make **property P**, with **n** values, the root of the current tree, this will **partition C into subsets**, **{C1, C2, . . . Cn}** and C_i denotes it. The expected information needed to complete the tree after making P the root is:

$$I(C) = \sum_{i=1}^{n} - p(C_i)\log_2(p(C_i)), \qquad E(P) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} I(C_i)$$

C : set of training instances, n : No. of value in property set P

- Information Gain calculates the **reduction in the entropy** and measures how well a given feature separates or classifies the target classes.

EXAMPLE

p(risk is high) = $^{6}/_{14}$, p(risk is moderate) = $^{3}/_{14}$, p(risk is low) = $^{5}/_{14}$

- I(C)= $$= -\frac{6}{14}\log_2\frac{6}{14} - \frac{3}{14}\log_2\frac{3}{14} - \frac{5}{14}\log_2\frac{5}{14}$$
$$= 1.531 \text{(bits)}$$

- if we make income the property tested at the root of the tree, this partitions the table of examples into the partitions C1 = {1,4,7,11}, C2 = {2,3,12,14}, and C3 = {5,6,8,9,10,13}. What is the expected information needed to complete the tree ?

EXAMPLE

$$E(\text{income}) = \frac{4}{14} I(C_1) + \frac{4}{14} I(C_2) + \frac{6}{14} I(C_3)$$

$$= \frac{4}{14} \times 0.0 + \frac{4}{14} \times 1.0 + \frac{6}{14} \times 0.650 = 0.564 \text{bits}$$

$$gain(\text{income}) = I(\text{table 9.1}) - E(\text{income})$$

$$= 1.531 - 0.564 = 0.967 \text{ bits}$$

$$gain(\text{credit history}) = 0.266$$

$$gain(\text{debt}) = 0.581$$

$$gain(\text{collateral}) = 0.756$$

I(C1)= I($0 to 15K)=-4/4* Log2 (4/4)-0-0=0
I(C2)= I($15 to 35K)= -2/4* Log2 (2/4)-2/4* Log2 (2/4)-0=½+½=1.0
I(C3)= I(over $35K)= -5/6* Log2 (5/6)-1/6* Log2 (1/6)-0=0.650

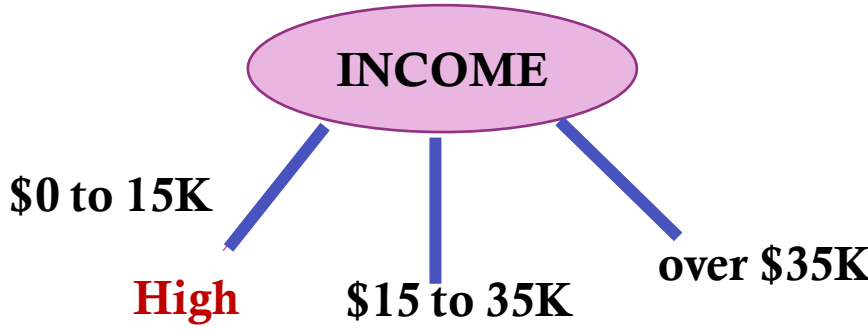
Because INCOME provides the greatest information gain, ID3 will select it as the root

EXAMPLE

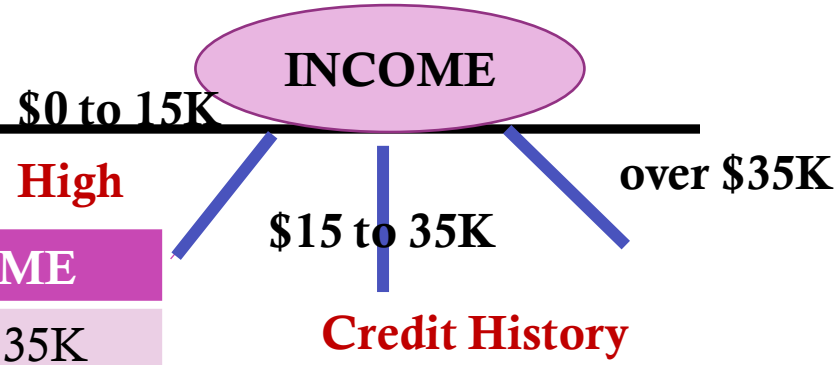
- what attribute should be tested at the Income branch node? Test dataset for custom subsets of **income attribute ie Credit History, Debit and Collateral**

NO	RISK	CREDIT HISTORY	DEBIT	COLLATERAL	INCOME
1	high	bad	high	none	$0 to 15K
2	high	unknown	low	none	$0 to 15K
3	high	bad	low	none	$0 to 15K
4	high	good	high	none	$0 to 15K

Risk is always **high** when income is **$0 to 15K**



EXAMPLE



NO	RISK	CREDIT HISTORY	DEBIT	COLLATERAL	INCOME
1	high	unknown	high	none	$15 to 35K
2	moderate	unknown	low	none	$15 to 35K
3	moderate	good	high	none	$15 to 35K
4	high	bad	high	none	$15 to 35K

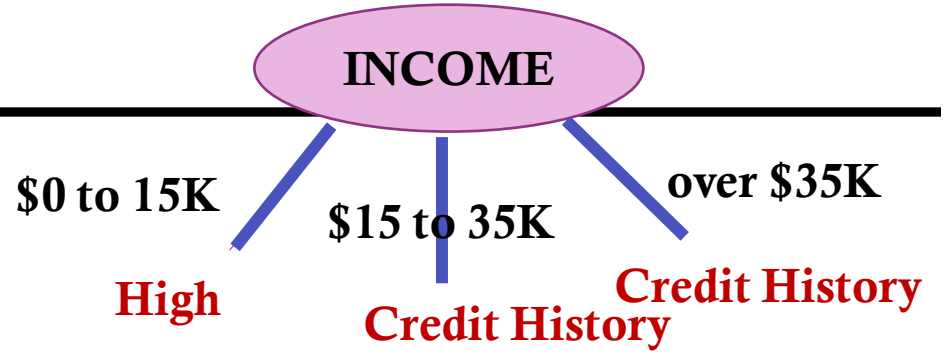
I(C1)= I(unknown)=-1/2* Log_2 (1/2)-1/2* Log_2 (1/2)=1
I(C2)= I(good)= 0
I(C3)= I(bad)= 0

I(table)=-2/4$\log_2$ 2/4-2/4$\log_2$ 2/4=0
Gain($P_{\$15 \text{ to } 35K}$, **Credit History**) = I(table)-E(Credit History)=-0.5
Gain($P_{\$15 \text{ to } 35K}$, **Debit**) = I(table)-E(Debit)=-1.188
Gain($P_{\$15 \text{ to } 35K}$, **Collateral**) = I(table)-E(Collateral)=-1
E(credit History)=2/4*I(C1)+¼*I(C2)+¼*I(C3)=0.5
E(Debit)=3/4*I(C1)+¼*I(C2)=1.188
E(Collateral)=4/4*I(C1)=1.
Here Credit History has highest gain

I(C1)= I(high)=-2/3* Log_2 (1/3)-1/3* Log_2 (1/3)=1.58
I(C2)= I(low)= -1/1*Log2(1/1)=0

I(C1)= I(none)=-2/4* Log_2 (2/4)-2/4* Log_2 (2/4)=1

EXAMPLE



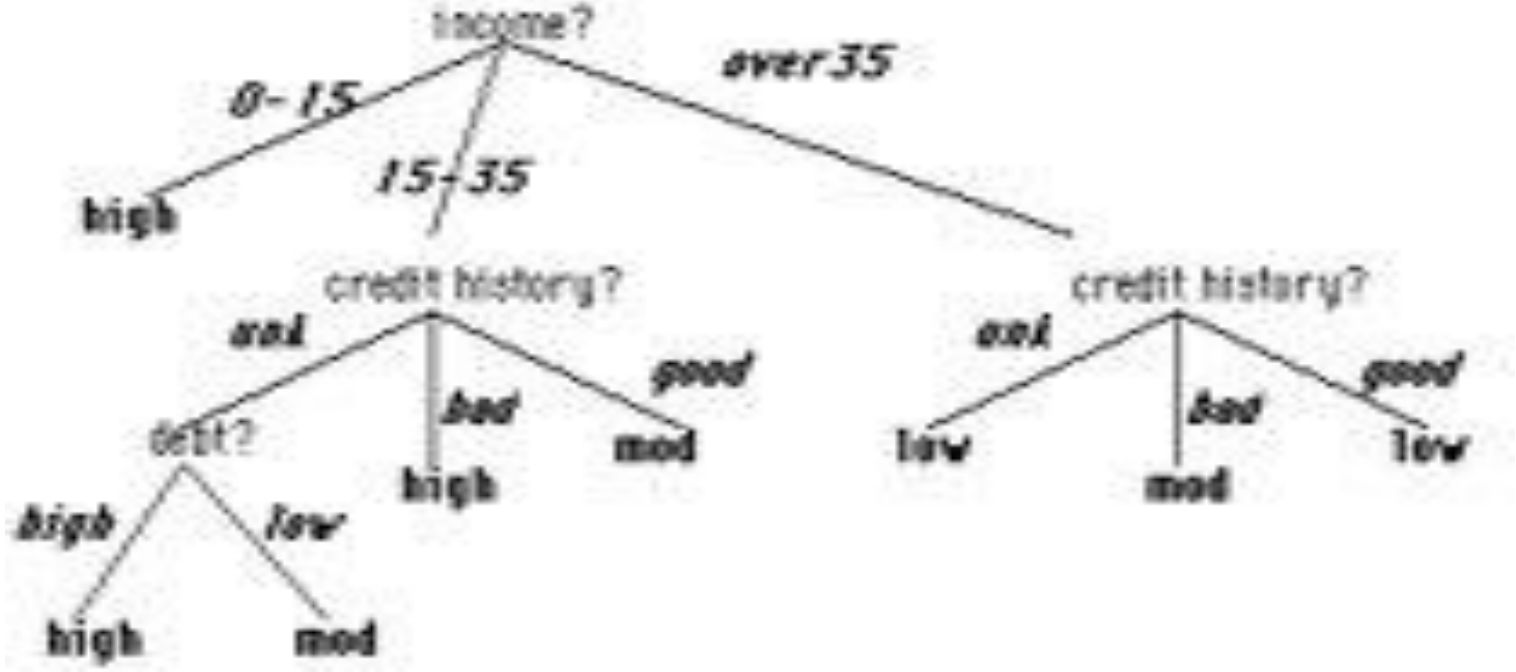
NO	RISK	CREDIT HISTORY	DEBIT	COLLATERAL	INCOME
1	low	unknown	low	none	over $35K
2	low	unknown	low	adequate	over $35K
3	moderate	bad	low	adequate	over $35K
4	low	good	low	none	over $35K
5	low	good	high	adequate	over $35K
6	low	good	high	none	over $35K

I(table)=-5/6*$\log_2$ 5/6-1/6*$\log_2$ 1/6=0.650
Gain($P_{\text{over \$35K}}$, **Credit History**) = I(table)-E(Credit History)=0.650-0
Gain($P_{\text{over \$35K}}$, **Debit**) = I(table)-E(Debit)=0.650-1.188
Gain($P_{\text{over \$35K}}$, **Collateral**) = I(table)-E(Collateral)=0.650-0.450
E(credit History)=2/6*I(C1)+6*I(C2)+3/6*I(C3)=0
E(Debit)=4/6*I(C1)+2/6*I(C2)=1.188
E(Collateral)=3/6*I(C1)+3/6*I(C2)=0.405
Here Credit History has highest gain

I(C1)= I(unknown)=-2/2* Log_2 (2/2)=0
I(C2)= I(good)= 0
I(C3)= I(bad)= 0

I(C1)= I(low)=-3/4* Log_2 (3/4)-1/4* Log_2 (1/4)=0.81
I(C2)= I(high)= -2/2*Log2(2/2)=0

EXAMPLE


income?
0-15
15-35
over35
high
credit history?
credit history?
bad
good
bad
good
debt?
high
mod
low
mod
high
low
high
mod

ID3 STEPS

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset **S** into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

INFORMATION GAIN IN TERMS OF ENTROPY

INFORMATION GAIN $\mathbf{IG(S, A) = Entropy(S) - \sum((|S_v| / |S|) * Entropy(S_v))}$

- where $\mathbf{S_v}$ is the set of rows in **S** for which the feature column **A** has value **v**, $\mathbf{|S_v|}$ is the number of rows in $\mathbf{S_v}$ and likewise $\mathbf{|S|}$ is the number of rows in **S.**
- **Entropy** is the measure of disorder and the Entropy of a dataset is the measure of disorder in the target feature of the dataset.

$$\mathbf{Entropy(S) = - \sum p_i * \log_2(p_i) \; ; \; i = 1 \text{ to } n}$$

- where,
  n is the total number of classes in the target column (in our case n = 2 i.e YES and NO)
  p_i is the **probability of class 'i'** or the ratio of "*number of rows with class i in the target column*" to the "*total number of rows*" in the dataset.
- In the case of binary classification (where the target column has only two types of classes) entropy is **0** if all values in the target column are homogenous(similar) and will be **1** if the target column has equal number values for both the classes.

EXAMPLE

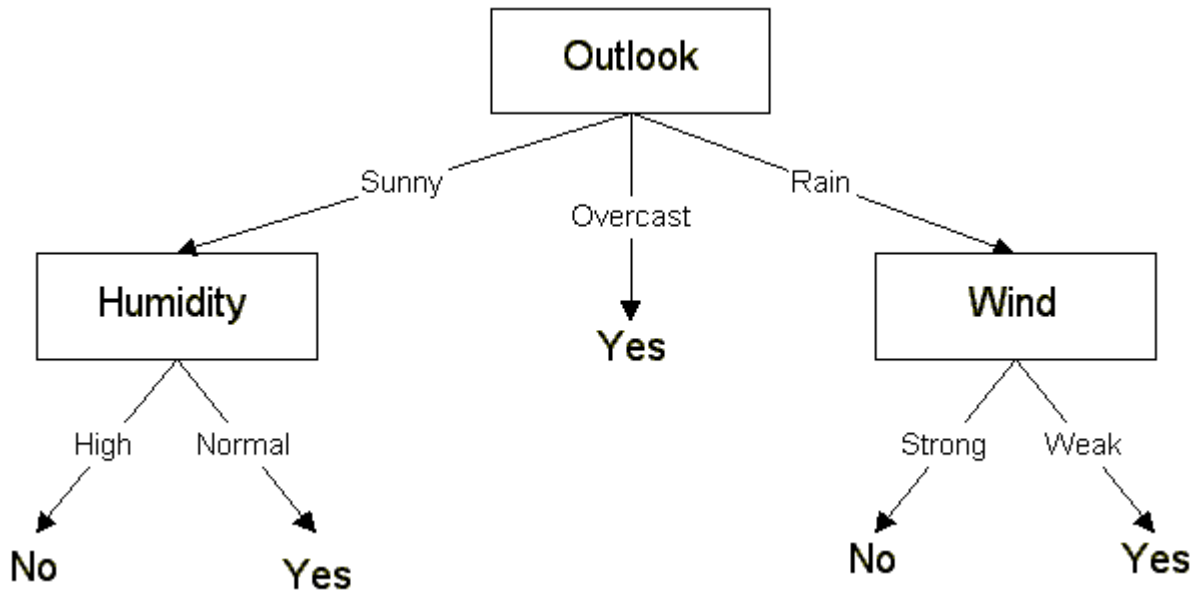
- Suppose S is a set of 14 examples in which one of the attributes is wind speed. The values of Wind can be *Weak* or *Strong*. The classification of these 14 examples are 9 YES and 5 NO. For attribute Wind, suppose there are 8 occurrences of Wind = Weak and 6 occurrences of Wind = Strong. For Wind = Weak, 6 of the examples are YES and 2 are NO. For Wind = Strong, 3 are YES and 3 are NO. Therefore
    - Gain(S,Wind)=Entropy(S)-(8/14)*Entropy(S_{weak})-(6/14)*Entropy(S_{strong})
- = 0.940 - (8/14)*0.811 - (6/14)*1.00
- = 0.048
- Entropy(S_{weak}) = - (6/8)*log2(6/8) - (2/8)*log2(2/8) = 0.811
- Entropy(S_{strong}) = - (3/6)*log2(3/6) - (3/6)*log2(3/6) = 1.00
- For each attribute, the gain is calculated and the highest gain is used in the decision node.

Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

CHALLENGE

- Suppose we want ID3 to decide whether the weather is amenable to playing baseball. Over the course of 2 weeks, data is collected to help ID3 build a decision tree (see table 1). The target classification is "should we play baseball?" which can be yes or no. The weather attributes are outlook, temperature, humidity, and wind speed. They can have the following values:
  - outlook = { sunny, overcast, rain }
  - temperature = {hot, mild, cool }
  - humidity = { high, normal }
  - wind = {weak, strong }
- If outlook is root node, then what attribute should be tested at the Sunny branch node?

TREE


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

ID3 STEPS

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset **S** into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

ID3 EVALUATION: ISSUES

- If data for some attribute is missing and is hard to obtain, it might be possible to extrapolate or use “unknown.”
- If some attributes have continuous values, groupings might be used.
- If the data set is too large, one might use bagging to select a sample from the training set. Or, one can use boosting to assign a weight showing importance to each instance. Or, one can divide the sample set into subsets and train on one, and test on others.

INDUCTIVE BIAS AND LEARNABILITY

- Usually the space of learning algorithms is very large
- Consider learning a classification of bit strings
  - A classification is simply a subset of all possible bit strings
  - If there are n bits there are 2^n possible bit strings
  - If a set has m elements, it has 2^m possible subsets
  - Therefore there are 2^(2^n) possible classifications (if n=50, larger than the number of molecules in the universe)
- We need additional heuristics (assumptions) to restrict the search space

INDUCTIVE BIAS AND LEARNABILITY

- Inductive bias refers to the assumptions that a machine learning algorithm will use during the learning process
- One kind of inductive bias is Occams Razor: assume that the simplest consistent hypothesis about the target function is actually the best
- Another kind is syntactic bias: assume a pattern defines the class of all matching strings
  - "nr" for the cards
  - {0, 1, #} for bit strings
- Note that syntactic bias restricts the concepts that can be learned
- If we use "nr" for card subsets, "all red cards except King of Diamonds" cannot be learned
- If we use {0, 1, #} for bit strings "1##0" represents {1110, 1100, 1010, 1000} but a single pattern cannot represent all strings of even parity (the number of 1s is even, including zero)
- The tradeoff between expressiveness and efficiency is typical

INDUCTIVE BIAS AND LEARNABILITY

- Some representational biases include
  - Conjunctive bias: restrict learned knowledge to conjunction of literals
  - Limitations on the number of disjuncts
  - Feature vectors: tables of observable features
  - Decision trees
  - Horn clauses
  - BBNs
- There is also work on programs that change their bias in response to data, but most programs assume a fixed inductive bias

KNOWLEDGE AND LEARNING

- **Similarity-based Learning**
- generalization is a function of similarities across training examples
- biases are limited to syntactic constraints on the form of learned knowledge
- **Knowledge-based Learning**
- the need of prior knowledge
- the most effective learning occurs when the learner already has considerable knowledge of the domain
- argument for the importance of knowledge
- similarity-based learning techniques rely on relatively large amount of training data. In contrast, humans can form reliable generalizations from as few as a single training instance.
- any set of training examples can support an unlimited number of generalizations, most of which are irrelevant or nonsensical.

KNOWLEDGE AND LEARNING

- **Explanation-Based Learning**
- use an explicitly represented domain theory to construct an explanations of a training example
- By generalizing from the explanation of theExplanation-Based Learning instance, EBL
  - filter noise
  - select relevant aspects of experience, and
  - organize training data into a systematic and coherent structure

Given

- A target concept
  - general specification of a goal state
- A training example
  - an instance of the target
- A domain theory
  - a set of rules and facts that are used to explain how the training example is an instance of the goal concept
- Operationality criteria
  - some means of describing the form that concept definitions may take

Determine

- A new schema that achieves target concept in a general way

EXAMPLE

- **We would like the system to learn what a cup is, i.e., we would like it to learn a rule of the form:** premise(X) $\rightarrow$ cup(X)

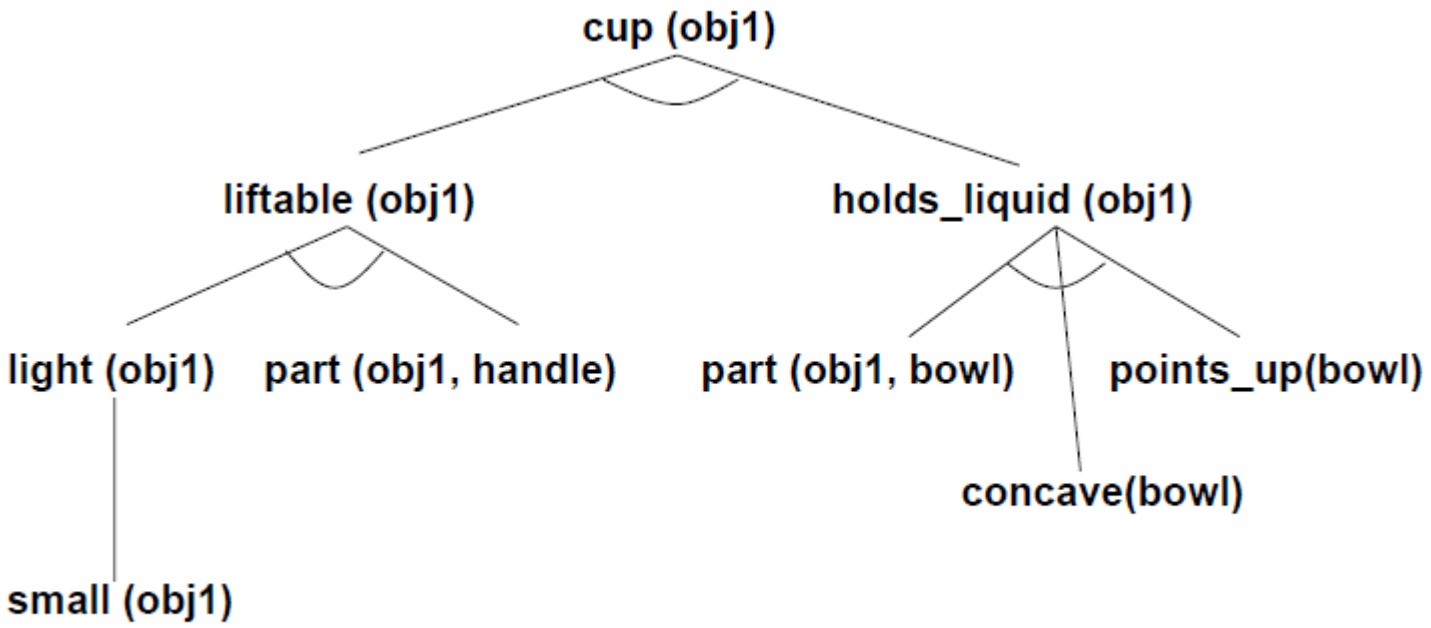
- **Assume that we have a domain theory:**

liftable(X) $\wedge$ holds_liquid(X) $\rightarrow$ cup(X)
part (Z,W) $\wedge$ concave(W) $\wedge$ points_up $\rightarrow$ holds_liquid (Z)
light(Y) $\wedge$ part(Y,handle) $\rightarrow$ liftable (Y)
small(A) $\rightarrow$ light(A)
made_of(A,feathers) $\rightarrow$ light(A)

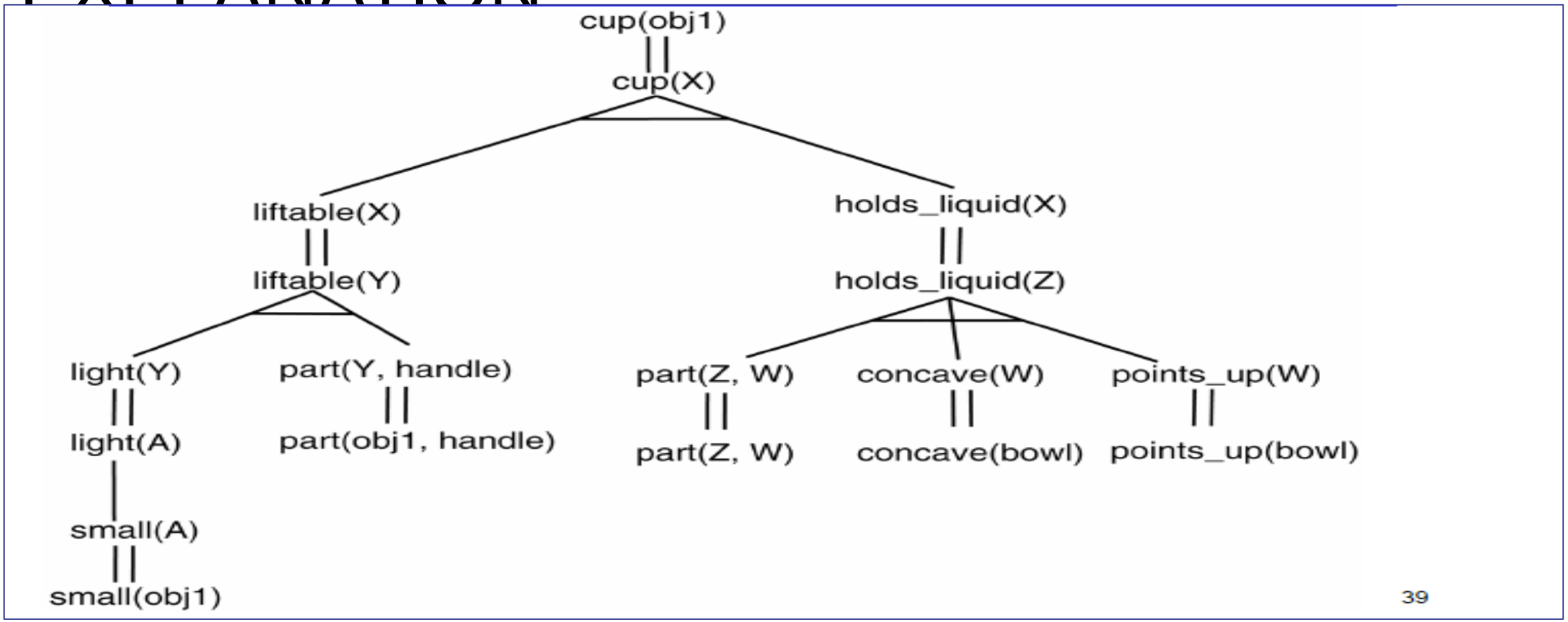
- **The training example is the following:**

cup (obj1)
small(obj1)
owns(bob,obj1)
part(obj1, bowl)
concave(bowl)
small(obj1)
part(obj1,handle)
part(obj1,bottom)
points_up(bowl)
color(obj1,red)

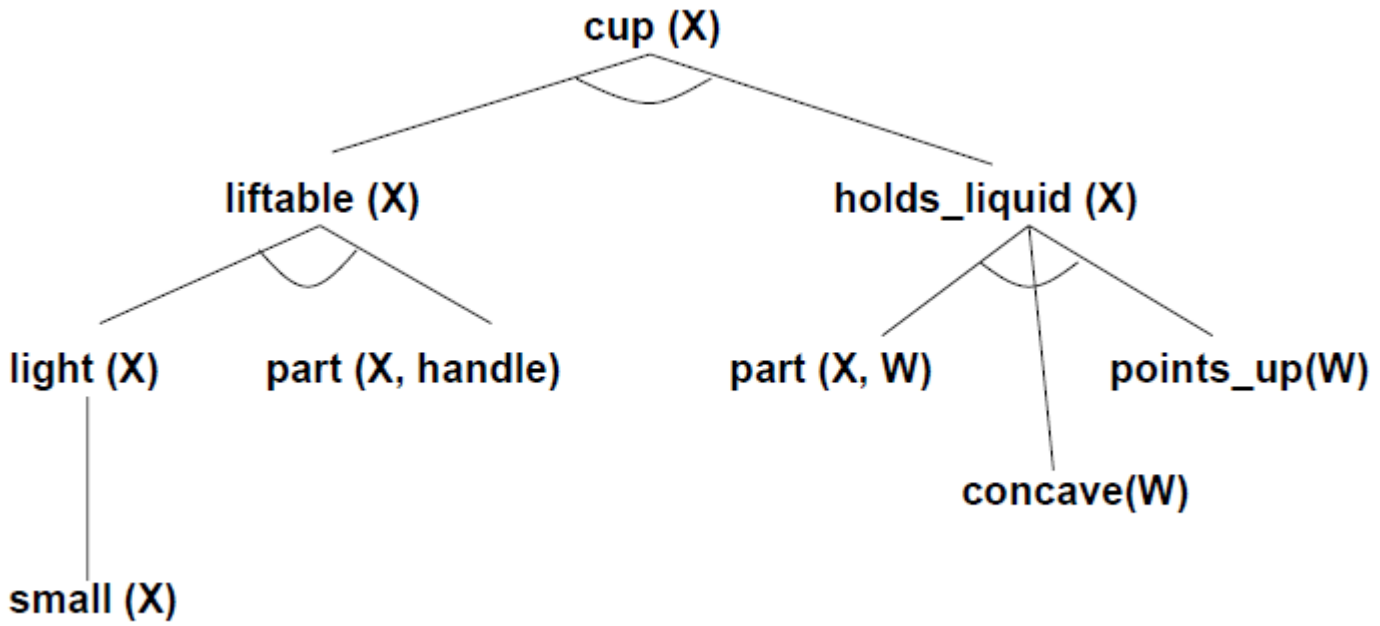
EXAMPLE: STEP 1: FIRST, FORM A SPECIFIC PROOF THAT OBJ1 IS A CUP



EXAMPLE STEP 2: ANALYZE THE EXPLANATION



EXAMPLE: STEP 3: ADOPT THE GENERALIZED THE PROOF



EBL ALGORITHM

- Initialize hypothesis = { }
- For each positive training example not covered by hypothesis:

1. Explain how training example satisfies target concept, in terms of domain theory

2. Analyze the explanation to determine the most general conditions under which this explanation (proof) holds

3. Refine the hypothesis by adding a new rule, whose premises are the above conditions, and whose consequent asserts the target concept

BENEFITS OF EBL

- select the relevant aspects of the training instance using the domain theory
- form generalizations relevant to specific goals and that are guaranteed to be logically consistent with the domain theory
- learning from single instance
- hypothesize unstated relationships between its goals and its experience by constructing an explanation

ISSUES IN EBL

Objection

- EBL cannot make the leaner do anything new
  - **EBL only learn rules within the deductive closure of its existing theory**
  - **sole function of training instance is to focus the theorem prover on relevant aspects of the problem domain**
  - **Viewed as a form of speed up learning or knowledge base reformation**

Responses to this objection

- Takes information implicit in a set of rules and makes it explicit
  - **E.g.) chess game**
- to focus on techniques for refining incomplete theories
  - development of heuristics for reasoning with imperfect theories, etc.
- to focus on integrating EBL and SBL.
  - EBL refine training data where the theory applies
  - SBL further generalize the partially generalized data

SUPERVISED VS UNSUPERVISED LEARNING

Supervised vs Unsupervised learning

Supervised learning

- the existence of a teacher, fitness function, some other external method of classifying training instances

Unsupervised learning

- eliminates the teacher
- requires that the learner form and evaluate concepts on its own.

UNSUPERVISED LEARNING

- Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.
- The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance.
- Common clustering algorithms include:
  - **Hierarchical clustering**: builds a multilevel hierarchy of clusters by creating a cluster tree
  - **k-Means clustering**: partitions data into k distinct clusters based on distance to the centroid of a cluster
  - **Gaussian mixture models**: models clusters as a mixture of multivariate normal density components
  - **Self-organizing maps**: uses neural networks that learn the topology and distribution of the data
  - **Hidden Markov models**: uses observed data to recover the sequence of states

CLUSTERING TYPES: CONCEPTUAL CLUSTERING

In this clustering method, Data are grouped in such a way that one data can belong to one cluster only.

Example: K-means

Given

- a collection of unclassified objects
- some means of measuring the similarity of objects

Goal

- organizing the objects into a classes that meet some standard of quality, such as maximizing the similarity of objects in a class

Numeric taxonomy

- The oldest approach to the clustering problem
- Represent a object as a collection of features (vector of n feature values)
- similarity metric : the euclidean distance between objects
- Build clusters in a bottom-up fashion

AGGLOMERATIVE CLUSTERING ALGORITHM

In this clustering technique, every data is a cluster. The iterative unions between the two nearest clusters reduce the number of clusters.

Example: Hierarchical clustering

step 1

- examine all pairs of objects
- select the pair with highest degree of similarity
- make the pair a cluster

step 2

- **define the features of the cluster as some function of the features of the component members**
- **replace the component objects with the cluster definition**

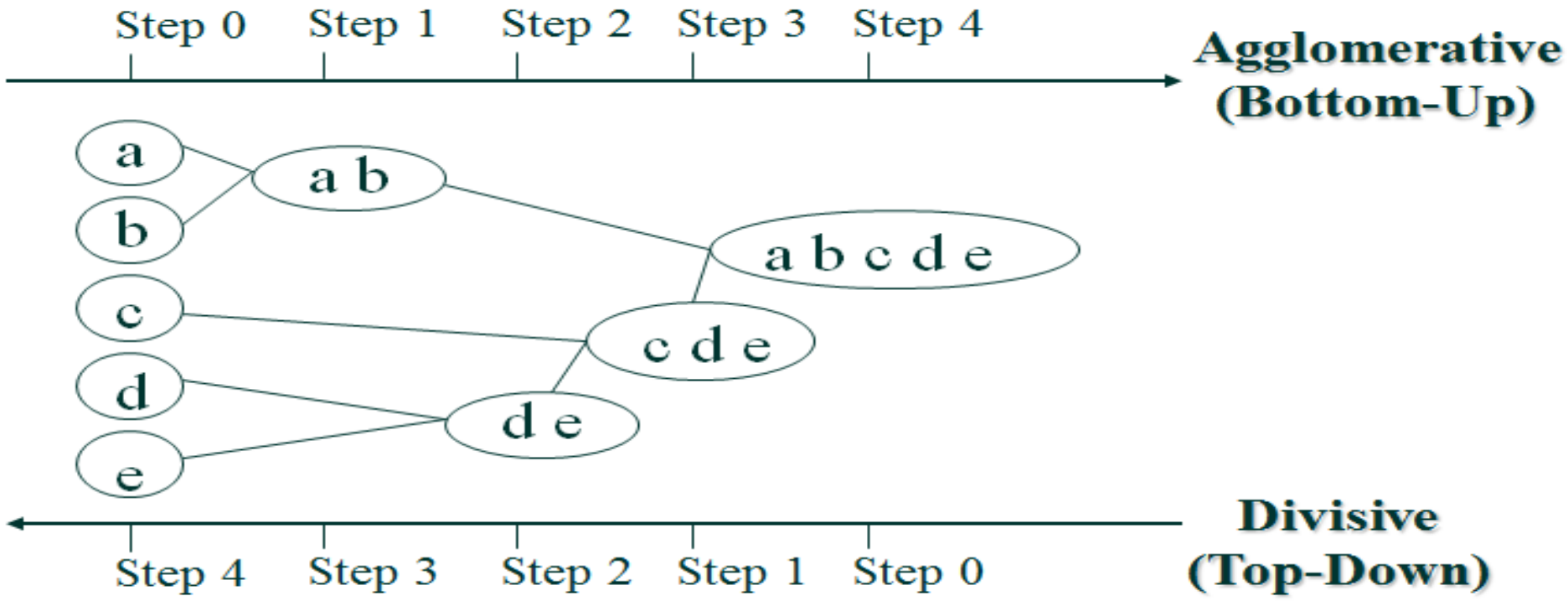
step 3

- **repeat the process on the collection of objects until all objects have been reduced to a single cluster**

- The result of the algorithm is a binary tree whose leaf nodes are instances and whose internal nodes are clusters of increasing size

HIERARCHICAL CLUSTERING

- **Agglomerative approach** vs. **Divisive approach**



- **Overlapping**
- In this technique, fuzzy sets is used to cluster data. Each point may belong to two or more clusters with separate degrees of membership.
- Here, data will be associated with an appropriate membership value. Example: Fuzzy C-Means
- **Probabilistic**
- This technique uses probability distribution to create the clusters

REINFORCEMENT LEARNING

- Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning
- Reinforcement Learning(RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.
- Unlike supervised learning where feedback provided to the agent is correct set of actions for performing a task, reinforcement learning uses rewards and punishment as signals for positive and negative behavior.
- The goal in unsupervised learning is to find similarities and differences between data points, in reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent.
- In reinforcement learning, we design computational algorithms for transforming world situations into actions in a manner that maximizes a reward measure. Our agent is not told directly what to do or which action to take; rather, the agent discovers through exploration which actions offer the most reward. Agents' actions affect not just immediate reward, but also impact subsequent actions and eventual rewards. These two features, trial-and-error search and delayed reinforcement, are the two most important characteristics of reinforcement learning.

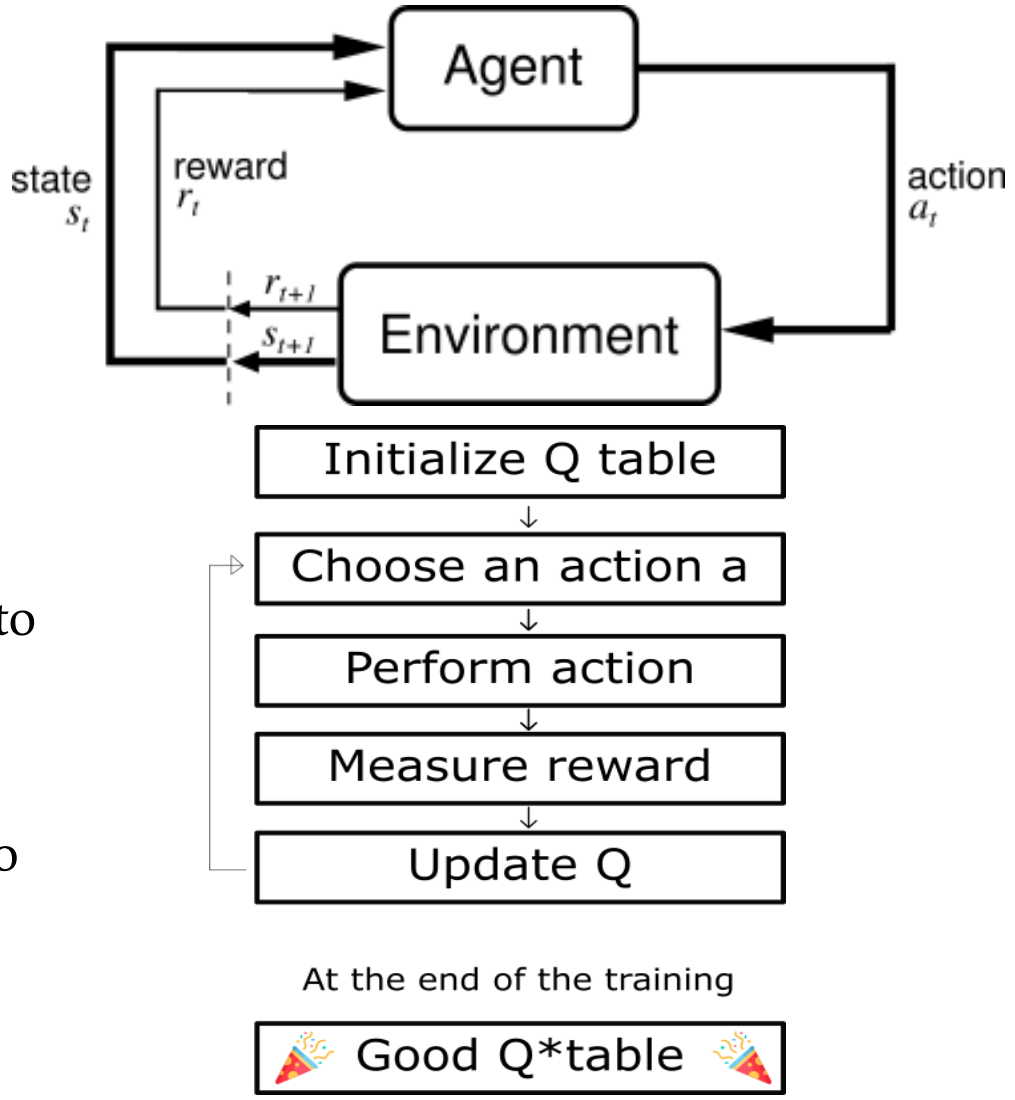
LEARNING MODELS OF REINFORCEMENT

1. **Markov Decision Process**

- A probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected determine a probability distribution on future states. Essentially, the outcome of applying an action to a state depends only on the current action and state

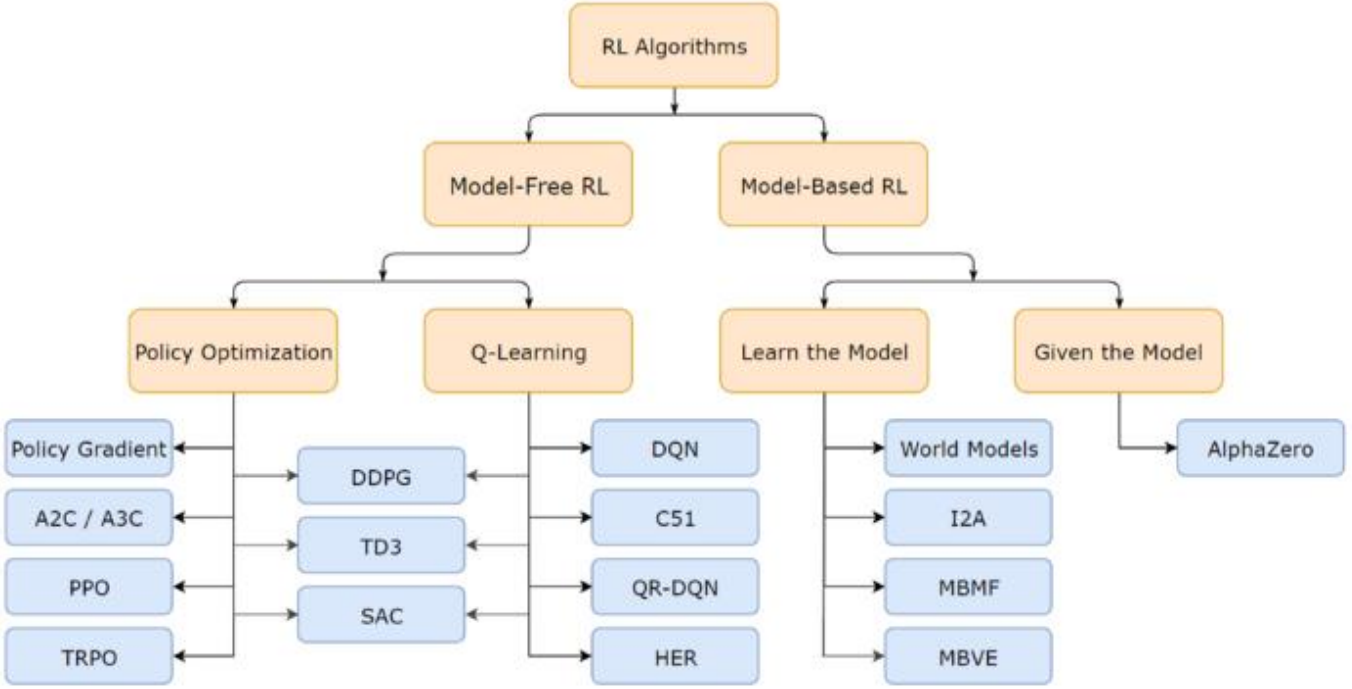
2. **Q-Learning**

Q learning is a value-based method of supplying information to inform which action an agent should take. Q-learning learns the action-value function *Q(s, a)*:



REINFORCEMENT LEARNING TAXONOMY

- *Model-based RL uses experience to construct an internal model of the transitions and immediate outcomes in the environment. Appropriate actions are then chosen by searching or planning in this world model.*
- *Model-free RL, on the other hand, uses experience to learn directly one or both of two simpler quantities (state/ action values or policies) which can achieve the same optimal behavior but without estimation or use of a world model.*



DIFFERENCE BETWEEN REINFORCEMENT LEARNING AND SUPERVISED LEARNING:

REINFORCEMENT LEARNING	SUPERVISED LEARNING
Reinforcement learning is all about making decisions sequentially. In simple words we can say that the output depends on the state of the current input and the next input depends on the output of the previous input	In Supervised learning the decision is made on the initial input or the input given at the start
In Reinforcement learning decision is dependent, So we give labels to sequences of dependent decisions	Supervised learning the decisions are independent of each other so labels are given to each decision.
Example: Chess game	Example: Object recognition

REFERENCES

- **Chapter 10:** Artificial Intelligence by George F Luger
- **Qlearning Example:** https://leonardoaraujosantos.gitbook.io/artificial-inteligence/artificial_intelligence/reinforcement_learning/qlearning_simple

THANK YOU

anooja@somaiya.edu