Inductive Learning (1/2)

Decision Tree Method

(If it's not simple,
it's not worth learning it)

R&N: Chap. 18, Sect. 18.1–3

1

Motivation

- An AI agent operating in a complex world requires an awful lot of knowledge: state representations, state axioms, constraints, action descriptions, heuristics, probabilities, ...
- More and more, AI agents are designed to acquire knowledge through learning

2

What is Learning?

- Mostly generalization from experience:

  "Our experience of the world is specific, yet we are able to formulate general theories that account for the past and predict the future"
  M.R. Genesereth and N.J. Nilsson, in *Logical Foundations of AI*, 1987
- → Concepts, heuristics, policies
- Supervised vs. un-supervised learning

3

Contents

- Introduction to inductive learning
- Logic-based inductive learning:
  - Decision-tree induction
- Function-based inductive learning
  - Neural nets

4

Logic-Based Inductive Learning

- Background knowledge KB
- Training set D (observed knowledge) that is not logically implied by KB
- **Inductive inference**: Find h such that KB and h imply D

h = D is a trivial, but un-interesting solution (data caching)

Rewarded Card Example

- Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded"
- Background knowledge KB:
  ((r=1) v … v (r=10)) ⇔ NUM(r)
  ((r=J) v (r=Q) v (r=K)) ⇔ FACE(r)
  ((s=S) v (s=C)) ⇔ BLACK(s)
  ((s=D) v (s=H)) ⇔ RED(s)
- Training set D:
  REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧ ¬REWARD([5,H]) ∧ ¬REWARD([J,S])

6

Rewarded Card Example

- Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded"
- Background knowledge KB:
  ((r=1) v … v (r=10)) ⇔ NUM(r)
  ((r=J) v (r=Q) v (r=K)) ⇔ FACE(r)
  ((s=S) v (s=C)) ⇔ BLACK(s)
  ((s=D) v (s=H)) ⇔ RED(s)
- Training set D:
  REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧ ¬REWARD([5,H]) ∧ ¬REWARD([J,S])
- Possible inductive hypothesis:
  h ≡ (NUM(r) ∧ BLACK(s) ⇔ REWARD([r,s]))

There are several possible inductive hypotheses

7

Learning a Predicate
(Concept Classifier)

- Set E of objects (e.g., cards)
- Goal predicate CONCEPT(x), where x is an object in E, that takes the value True or False (e.g., REWARD)

Example:
CONCEPT describes the precondition of an action, e.g., Unstack(C,A)
- E is the set of states
- CONCEPT(x) ⇔
  HANDEMPTY∈x, BLOCK(C) ∈x, BLOCK(A) ∈x, CLEAR(C) ∈x, ON(C,A) ∈x

Learning CONCEPT is a step toward learning an action description

8

Learning a Predicate
(Concept Classifier)

- Set E of objects (e.g., cards)
- Goal predicate CONCEPT(x), where x is an object in E, that takes the value True or False (e.g., REWARD)
- Observable predicates A(x), B(X), … (e.g., NUM, RED)
- Training set: values of CONCEPT for some combinations of values of the observable predicates

9

Example of Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

10

Example of Training Set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overc…		High	Weak	Yes
D4	Rai…		High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
				…g	Yes
				…k	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Ternary attributes

Goal predicate is PLAY-TENNIS

Note that the training set does not say whether an observable predicate is pertinent or not

11

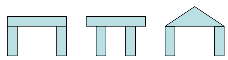
Learning a Predicate
(Concept Classifier)

- Set E of objects (e.g., cards)
- Goal predicate CONCEPT(x), where x is an object in E, that takes the value True or False (e.g., REWARD)
- Observable predicates A(x), B(X), … (e.g., NUM, RED)
- Training set: values of CONCEPT for some combinations of values of the observable predicates
- Find a representation of CONCEPT in the form:
  CONCEPT(x) ⇔ S(A,B, …)
  where S(A,B,…) is a sentence built with the observable predicates, e.g.:
  CONCEPT(x) ⇔ A(x) ∧ (¬B(x) v C(x))

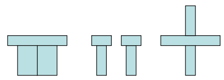
12

Learning an Arch Classifier

- These objects are arches: (positive examples)



- These aren't: (negative examples)



ARCH(x) ⇔ HAS-PART(x,b1) ∧ HAS-PART(x,b2) ∧
HAS-PART(x,b3) ∧ IS-A(b1,BRICK) ∧
IS-A(b2,BRICK) ∧ ¬MEET(b1,b2) ∧
(IS-A(b3,BRICK) v IS-A(b3,WEDGE)) ∧
SUPPORTED(b3,b1) ∧ SUPPORTED(b3,b2)

Example set

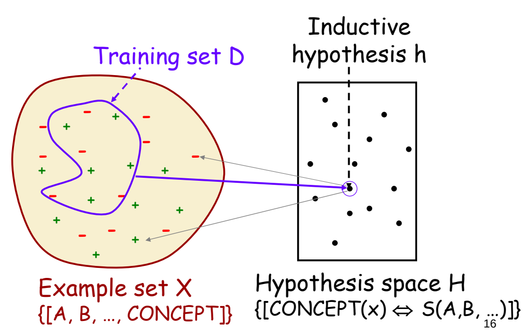
- An example consists of the values of CONCEPT and the observable predicates for some object x
- An example is positive if CONCEPT is True, else it is negative
- The set X of all examples is the example set
- The training set is a subset of X

a small one!

Hypothesis Space

- An hypothesis is any sentence of the form:
  CONCEPT(x) ⇔ S(A,B, …)
  where S(A,B,…) is a sentence built using the observable predicates
- The set of all hypotheses is called the hypothesis space H
- An hypothesis h agrees with an example if it gives the correct value of CONCEPT

Inductive Learning Scheme



Size of Hypothesis Space

- n observable predicates
- 2^n entries in truth table defining CONCEPT and each entry can be filled with True or False
- In the absence of any restriction (bias), there are 2^{2^n} hypotheses to choose from
- n = 6 → 2×10^{19} hypotheses!

Multiple Inductive Hypotheses

- Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded"
- Background knowledge KB:
  ((r=1) v … v (r=10)) ⇔ NUM(r)
  ((r=J) v (r=Q) v (r=K)) ⇔ FACE(r)
  ((s=S) v (s=C)) ⇔ BLACK(s)
  ((s=D) v (s=H)) ⇔ RED(s)
- Training set D:
  REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧ ¬REWARD([5,H]) ∧ ¬REWARD([J,S])

h_1 ≡ NUM(r) ∧ BLACK(s) ⇔ REWARD([r,s])
h_2 ≡ BLACK(s) ∧ ¬(r=J) ⇔ REWARD([r,s])
h_3 ≡ ([r,s]=[4,C]) ∨ ([r,s]=[7,C]) ∨ [r,s]=[2,S]) ⇔ REWARD([r,s])
h_4 ≡ ¬([r,s]=[5,H]) ∨ ¬([r,s]=[J,S]) ⇔ REWARD([r,s])
agree with all the examples in the training set

Multiple Inductive Hypotheses

- Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded"

Need for a system of preferences – called a **bias** – to compare possible hypotheses

((s=D) v (s=H)) ⇔ RED(s)

- Training set D:
REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧ ¬REWARD([5,H]) ∧ ¬REWARD([J,S])

$h_1 \equiv$ NUM(r) ∧ BLACK(s) ⇔ REWARD([r,s])
$h_2 \equiv$ BLACK(s) ∧ ¬(r=J) ⇔ REWARD([r,s])
$h_3 \equiv$ ([r,s]=[4,C]) ∨ ([r,s]=[7,C]) ∨ [r,s]=[2,S]) ⇔ REWARD([r,s])
$h_4 \equiv$ ¬([r,s]=[5,H]) ∨ ¬([r,s]=[J,S]) ⇔ REWARD([r,s])
agree with all the examples in the training set

19

Notion of Capacity

- It refers to the ability of a machine to learn any training set without error
- A machine with too much capacity is like a botanist with photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything he has seen before
- A machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree
- Good generalization can only be achieved when the right balance is struck between the accuracy attained on the training set and the capacity of the machine

20

→ Keep-It-Simple (KIS) Bias

- **Examples**
  - Use much fewer observable predicates than the training set
  - Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax
- **Motivation**
  - If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
  - There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

21

→ Keep-It-Simple (KIS) Bias

- **Examples**
  - Use much fewer observable predicates than the training set
  - Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to have simple syntax
- **M**

Einstein: "A theory must be as simple as possible, but not simpler than this"

  - If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
  - There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

22

→ Keep-It-Simple (KIS) Bias

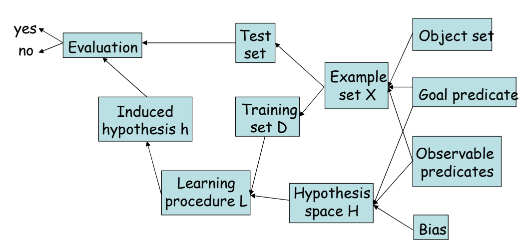
- **Examples**

If the bias allows only sentences S that are conjunctions of k ‹‹ n predicates picked from the n observable predicates, then the size of H is $O(n^k)$

- **Motivation**
  - If an hypothesis is too complex it is not worth learning it (data caching does the job as well)
  - There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

23

Putting Things Together



24

Decision Tree Method

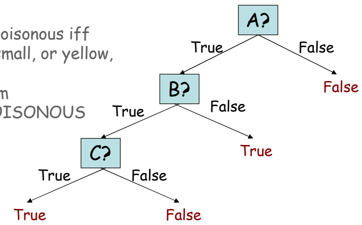
25

Predicate as a Decision Tree

The predicate CONCEPT(x) ⇔ A(x) ∧ (¬B(x) v C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED



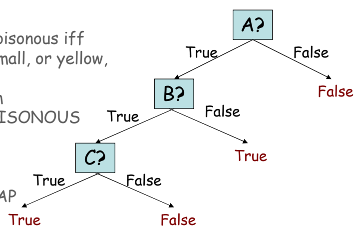
26

Predicate as a Decision Tree

The predicate CONCEPT(x) ⇔ A(x) ∧ (¬B(x) v C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED
- D = FUNNEL-CAP
- E = BULKY



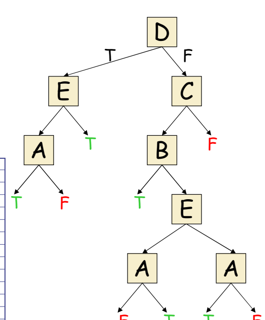
27

Training Set

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

28

Possible Decision Tree



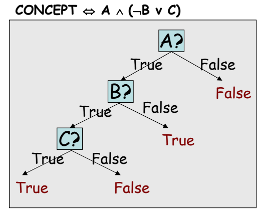
Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

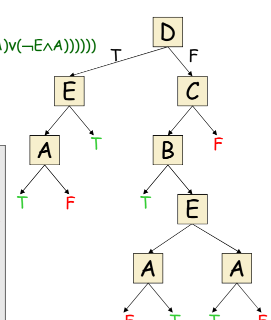
29

Possible Decision Tree

CONCEPT ⇔
(D∧(¬E∨A))∨(¬D∧(C∧(B∨(¬B∧((E∧¬A)∨(¬E∧A))))))

CONCEPT ⇔ A ∧ (¬B v C)

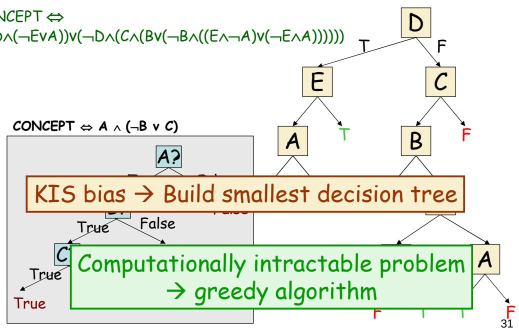




30

Possible Decision Tree

CONCEPT ⇔ $(D \wedge (\neg E \vee A)) \vee (\neg D \wedge (C \wedge (B \vee (\neg B \wedge ((E \wedge \neg A) \vee (\neg E \wedge A))))))$



KIS bias → Build smallest decision tree

Computationally intractable problem → greedy algorithm

Getting Started:
Top-Down Induction of Decision Tree

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

Getting Started:
Top-Down Induction of Decision Tree

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (**majority rule**) with an estimated probability of error P(E) = 6/13

Assuming that we will only include one observable predicate in the decision tree, **which predicate should we test to minimize the probability of error (i.e., the # of misclassified examples in the training set)?** → Greedy algorithm

Assume It's A

A
T F

	T	F
True:	6, 7, 8, 9, 10, 13	
False:	11, 12	1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise

→ The number of misclassified examples from the training set is 2

Assume It's B

B
T F

	T	F
True:	9, 10	6, 7, 8, 13
False:	2, 3, 11, 12	1, 4, 5

If we test only B, we will report that CONCEPT is False if B is True and True otherwise

→ The number of misclassified examples from the training set is 5

Assume It's C

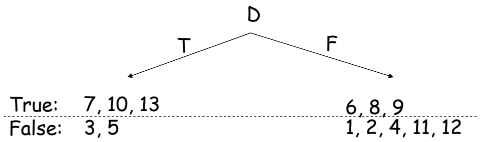
C
T F

	T	F
True:	6, 8, 9, 10, 13	7
False:	1, 3, 4	1, 5, 11, 12

If we test only C, we will report that CONCEPT is True if C is True and False otherwise

→ The number of misclassified examples from the training set is 4

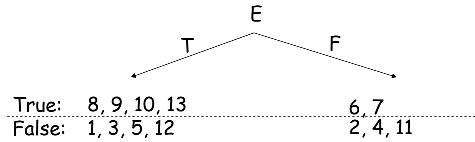
Assume It's D



If we test only D, we will report that CONCEPT is True if D is True and False otherwise

→ The number of misclassified examples from the training set is 5

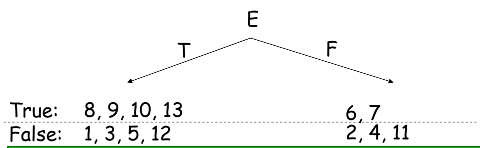
Assume It's E



If we test only E we will report that CONCEPT is False, independent of the outcome

→ The number of misclassified examples from the training set is 6

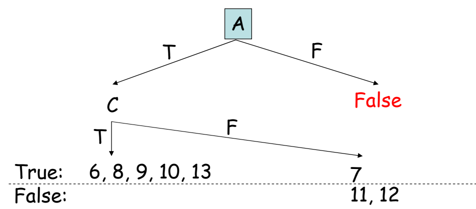
Assume It's E



So, the best predicate to test is A e,
independent of the outcome

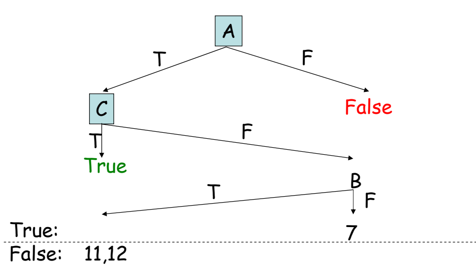
→ The number of misclassified examples from the training set is 6

Choice of Second Predicate

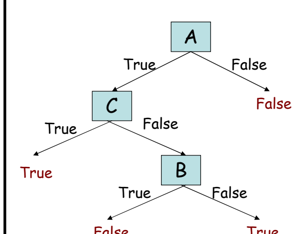


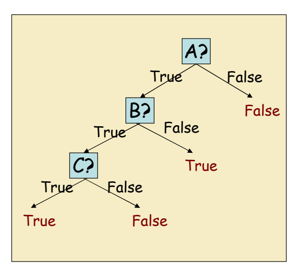
→ The number of misclassified examples from the training set is 1

Choice of Third Predicate



Final Tree

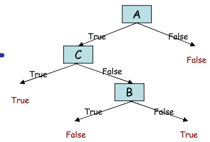




CONCEPT ⇔ A ∧ (C v ¬B)

CONCEPT ⇔ A ∧ (¬B v C)

Top-Down Induction of a DT

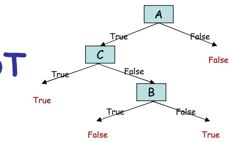


DTL(Δ, Predicates)

1. If all examples in Δ are positive then return True
2. If all examples in Δ are negative then return False
3. If Predicates is empty then return *failure*
4. A ← error-minimizing predicate in Predicates
5. Return the tree whose:
   - root is A,
   - left branch is DTL(Δ^{+A},Predicates-A),
   - right branch is DTL(Δ^{-A},Predicates-A)

Subset of examples that satisfy A

Top-Down Induction of a DT



DTL(Δ, Predicates)

1. If all examples in Δ are positive then return True
2. If all examples in Δ are negative then return False
3. If Predicates is empty then return *failure*
4. A ← error-minimizing predicate in Predicates
5. Return the tree whose:
   - root is A,
   - left branch is DTL(Δ^{+A},Predicates-A),
   - right branch is DTL(Δ^{-A},Predicates-A)

Noise in training set!
May return majority rule,
instead of failure

Comments

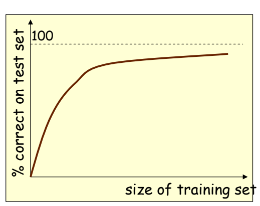
- Widely used algorithm
- Greedy
- Robust to noise (incorrect examples)
- Not incremental

Using Information Theory

- Rather than minimizing the probability of error, many existing learning procedures minimize the expected number of questions needed to decide if an object x satisfies CONCEPT
- This minimization is based on a measure of the "quantity of information" contained in the truth value of an observable predicate
- See R&N p. 659-660

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve

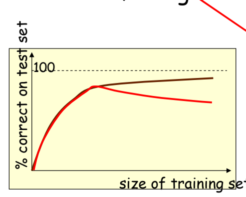


Typical learning curve

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting

Risk of using irrelevant observable predicates to generate an hypothesis that agrees with all examples in the training set



Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting
  - Tree pruning

Risk of using irrelevant observable predicates to generate an hypothesis that agrees with all examples in the training set

Terminate recursion when # errors / information gain is small

49

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting
  - Tree pruning

Risk of using irrelevant observable predicates to ...sis ...mples

The resulting decision tree + majority rule may not classify correctly all examples in the training set

Terminate recursion when # errors / information gain is small

50

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting
  - Tree pruning
- Incorrect examples
- Missing data
- Multi-valued and continuous attributes

51

Applications of Decision Tree

- Medical diagnostic / Drug design
- Evaluation of geological systems for assessing gas and oil basins
- Early detection of problems (e.g., jamming) during oil drilling operations
- Automatic generation of rules in expert systems

52