Data mining, machine learning, and uncertainty reasoning

林偉川

Search through the space of generalization

- A representation language has been selected by learners to learn a concept from a set of positive and negative examples (space of generalization)
- If the descriptions are based on attribute-value logic, the space of all concepts is large → Ten attributes with five possible values for each of them amount to $5^{10}=9765625$ possible vectors

Search through the space of generalization

- Any subset of such vectors can correspond to a concept, which means $2^{9765625}$ concepts can be defined over these attributes
- Background knowledge can limit the size of the representation space
- To cope with the problem of computational tractability, the learner combines two powerful techniques: induction and heuristic search

3

Inductive Essence of learning

- An ET has some preliminary linguistic knowledge and asks "what is a bird?"
- A blackbird is a positive example of the concept. However, it is a hard job to teach it. (S)
- To memorizing all features of blackbirds is hardly sufficient to recognize other birds as instances of the same category ➔ A generalization of this example is needed.

4

Inductive Essence of learning

- A negative example is what is not a bird? ➔ (G) a dog is not a bird because it do not possess wings
- All creatures with wings are birds ➔ too general
- A fly is this category but not a bird. A specialization is necessary.
- A noticable features of the blackbird → absent in dogs and flies is that Birds have beaks
- Finally, ET concludes birds are winged creature with beaks

5

Inductive Essence of learning

- The set of the most specific descriptions, denoted by S
- The most general descriptions denoted by G
- The G has to be specialized
- The next positive examples should enriches the set S with another most specific description
- Generalization is applied to the set S whenever a new positive example arrives
- A negative example can necessitate the specialization of the set G ➔ version space algorithm

6

Inductive Essence of learning

- Version space algorithm built on the idea of gradual reduction of the space of current versions of the concept description
- Concept learning can be viewed as a series of generalization and specialization of a single hypothesis
- Concept learning can also be conceived as a search through the space of descriptions, the essential search operators are generalization and specialization

7

Inductive Essence of learning

- Horn clauses can be generalized by turning a constant into a variable or by dropping a condition

  P(x, y) :- q(x,2), r(y,2) $\rightarrow_G$ P(x, y) :- q(x, z), r(y, z) $\rightarrow_G$ P(x, y) :- q(x, 2)

- A Horn clause can be specialized by turning a variable into a constant, or by adding a literal to the clause
- Proper selection of the search operators is the critical task of the designer of a learning program

8

Search process

- A widespread framework for concept learning is search through the space of descriptions permitted by the learner's representation language
- Search techniques have been widely investigated

Search process

A search process explores states in a search space according to the following steps:

1. Initial state: the starting position of the search (the most specific concept description ➔ positive example)
2. Termination criterion: the objective to be arrived at. States that satisfy the termination criterion are referred as final states (covers all positive and no negative examples)

Search process

3. Search operators advance the search from one state to another (operators are generalizations and/or specialization of concept descriptions)
4. Search strategy determines under what conditions and to which state an operator is to be applied

- There are two fundamental systematic searches such as depth-first and breadth-first search
- Visualize the space of all possible states as an oriented graph whose nodes represent individual states and the edges are the search operator

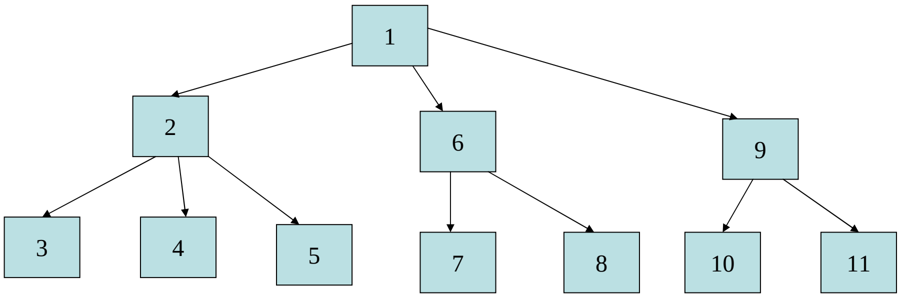
11

Depth-first search

- An operator is applied to the initial state S_1, arriving at a new state S_2. If S_2 is not the final states, then, again, an operator is applied to S_2 arriving at a new state S_3
- If no new state can be reached in this way and the final state has not been found, the system backtracks to the previous state and applies some other operator
- If this is not possible, the system backtracks until a state is found that allows the application of some of the operators.
- If no such state, the search terminates

12

Depth-first search



13

Breadth-first search

- The numbers in the rectangles indicate the order in which the states are visited
- All operators are applied, one by one, to the initial state S_1, the result states are tested. If some of them are the final states, the search algorithm stops
- Otherwise, the operators are applied to all subsequence states, then again to the subsequence states, and so on, until the termination criterion is satisfied

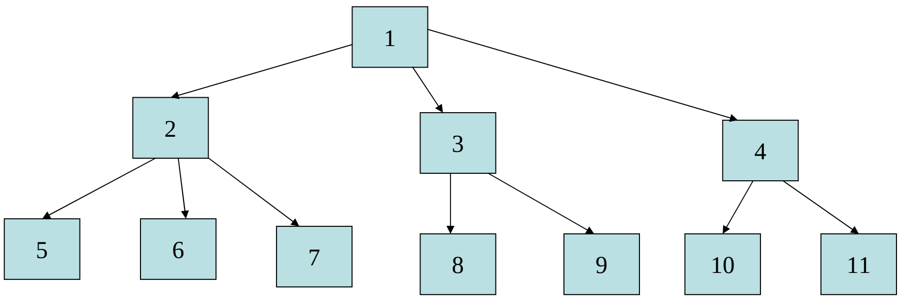
14

Breadth-first search

- BFS assumes no backtracking, which is a slight simplification of the task than DFS
- However, the searcher must store many intermediate states
- Time V.S. Space waste!!! ➔ trade-off

15

Breadth-first search



16

Heuristic Search

- Decide which of the available operator will lead to the closest proximity of the final state
- This requires an evaluation function to assess the value of each of the states reached ➔ assume the evaluation function is given
- Two search algorithms are of this types such as BEST-FIRST, BEAM search algorithm

17

BEST-FIRST search algorithm

1. Let the initial state be referred to as the best state, the set of current states consist of this single state
2. If the best state satisfies the given termination criterion, then stop ➔ the best state is the solution of the search
3. Apply all applicable operators to the best state, thus creating a set of new states that are added to the set of current states

18

BEST-FIRST search algorithm

4. Evaluate all current states. Decide which is the best state and go to step 2.

➔ Differs from the BFS in that it always extends only the most promising state, thus speeding up the search

➔ The price is the danger of falling to a local maximum of the evaluation function

19

BEST-FIRST Search process

- ⊕ stands for positive example and ⊗ stands for negative example
- 2 operators for the example to demonstrate the search process：
  - specialize the current description by adding a conjunction
  - generalize the current description by adding a disjunction

20

BEST-FIRST Search process

- Initial state is any description. The application of the specialization operator will produce the descriptions: at1=a, at1=b, at2=x, at2=y, at2=z, at3=m, and at3=n
- At2=x and at3=m do not cover any ⊗ and will probably achieve the highest value of a reasonable evaluation (the row in red)

21

BEST-FIRST Search process

- For some reason, at2=x is preferred and will become the best description
- Some ⊕'s in the table are now not covered, the learner will try to apply the search operator to the best description
- Applying the generalize operator → at2=x V at2=y, the number of ⊕s covered increased and the evaluation function confirms this description is better than at2=x

22

BEST-FIRST Search process

- The new description covers all ⊕s but it also covers 2 ⊗s.
- The description is specialized into at2=x V 【(at2=y) Λ X】, where X stands for any of the following conjuncts : at1={a, b} Λ at3={m} and at3={n}
  - Among the new states ,the best one is at2=x V 【(at2=y) Λ (at3=m)】. As it covers all ⊕s but no ⊗s, the search terminates
  - The best-first search requires excessive memory because it stores all generated states

23

Car can attract analysis?

Object	Make	Size	Price	Classification
Car1	European	Big	Affordable	⊕ (positive)
Car2	Japanese	Big	Affordable	⊕
Car3	European	Medium	Affordable	⊗ (negative)
Car4	European	Small	Affordable	⊗
Car5	European	Medium	Expensive	⊕
Car6	Japanese	Medium	Affordable	⊗
Car7	Japanese	Medium	Expensive	⊕
Car8	European	Big	Expensive	⊕

24

Positive and negative examples for concept learning

example	at1	at2	at3	Classification
e1	a	x	n	$\oplus$ (positive)
e2	b	x	n	$\oplus$
e3	a	y	n	$\otimes$ (negative)
e4	a	z	n	$\otimes$
e5	a	y	m	$\oplus$
e6	b	y	n	$\otimes$
e7	b	y	m	$\oplus$
e8	a	x	m	$\oplus$

Beam Search Algorithm

- A more economic approach is the beam search that only contains N best states at any time

Algorithm of Beam search Algorithm：

1. Let the initial state be the best state
2. If the best state satisfies some termination criterion, then stop ➔ the best state is the solution of the search
3. If the number of current states is larger than N, keep only the N best states and delete all others

Beam Search Algorithm

4. Apply the search operators to the best state, and add the newly created states to the set of current states
5. Evaluate all states and go to step 2

⌘ A popular instantiation of the beam-search algorithm is defined N=1 is sometimes called hill-climbing algorithm.

⌘ Hill climbers striving to find the shortest trajectory to the peak always pick the steepest path

27

Classic methods of learning

2 essential learning principles

- Divide-and-Conquer
  - The entire set of examples is split into subsets that are more easy to handle (TDIDT algorithm)
- AQ-philosophy
  - Based on the idea of progressive coverage of the training data by consecutively generated decision rules

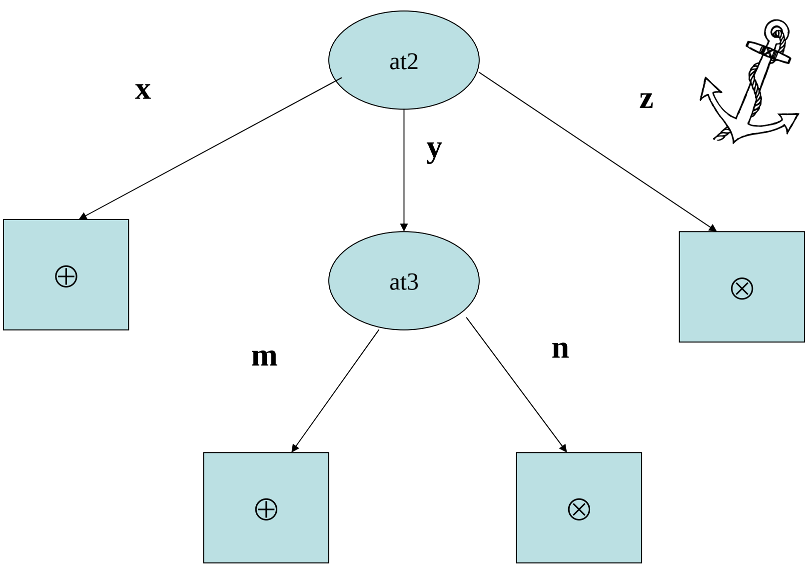
28

Divide-and-Conquer Learning

- In attributional logic, the partitioning is carrying out along attribute values so that all examples in a subset share the same value of given attribute
- Table is analyzed to classify as at1={a, b}， at2={x, y, z}， at3={m, n} (solutions is at2=x V 【(at2=y) Λ (at3=m)】)
- Induction of decision trees is known under the acronym TDIDT (Top-Down Induction of Decision Tree) or ID3.

29

Decision tree of example table



30

Decision tree explanation

- e1 has at2=x which sends it downward along the leftmost branch, only to end up in the box labeled with ⊕
- e3 has at2=y and at3=n, which is end up in the box labeled with ⊗

31

Decision tree explanation

These tree can be rewritten as following logical expressions:

- (class= ⊕) ← (at2=x) V 【(at2=y) Λ (at3=m)】
- (match solutions ： at2=x V 【(at2=y) Λ (at3=m)】)
- (class= ⊗) ← (at2=z) V 【(at2=y) Λ (at3=n)】
- The classification of examples that do not satisfy either of these rules can be based on the distance between the example description and the rules or an “I-don’t-know” answer can be issued

32

Homework

- Describe the table in P27 in the conjunction rules
- Draw the possible decision tree of this example!!