L1 Intro

2024年9月20日 20:02

L2 Symbolic AI

2024年9月20日 20:03

State Space Representation:
used to model a question

Predicate calculus:谓词演算，用于计算机的knowledge representation
Graph:
Numerical AI: NN

We mainly use the GRAPH, especially Tree.

L3 State Space Search

2024年9月20日 20:03

[N, A, S, GD]

Node
Arc
Start
Goal

- Search strategies: data/goal-driven

L4 Systematically Search

2024年9月20日 20:03

First Search:

Notations of *Backtracking*

- ***CS*** = *Current State* (the state currently under consideration)
- ***SL*** = *State List* (the list of states in the current path being pursued. If a goal is found, SL contains the ordered list of states on the **solution path**)
- ***NSL*** = *New State List* (the list of new states contains nodes awaiting evaluation, i.e., nodes whose descendants have not yet been generated and searched) (Unprocessed states)
- ***DE*** = *Dead Ends* (the list of states whose descendants have failed to contain a goal node. If these states are encountered again, they will be deleted as elements of *DE* and eliminated)
- ***CS*** (Current State) is always equal to the state most recently added to *SL* and represents the "frontier" of the solution path currently being explored

Backtracking: meet goal, return. meet badend, backtrack.

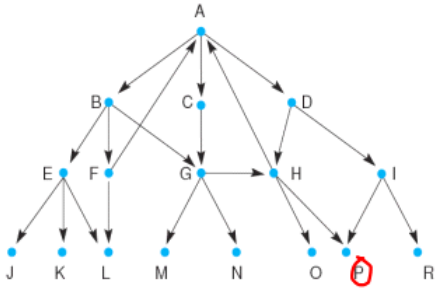
Exercise3

remember add to left(stack)

LIFO&FILO

Exercise 3: The BackTracking

- "Hand run" the backtracking algorithm on the right graph
- Keep track of the successive values of NSL, SL, CS, DE.
- We assume there is no goal state in the search space.
- Initialize: SL=[A];NSL=[A]; DE=[]; CS=A;



Loop	*CS*	*SL*	*NSL*	*DE*
0	A	[A]	[A]	//check CS =goal? Has new children ?
1	B	[B,A]	[**B,C,D**,A]	//add children to NSL – 'stack'; reset CS; update SL
2	E	[E,B,A]	[**E,F,G**, B,C,D,A]	//add children to NSL – 'stack'; reset CS; update SL
3	J	[J, E,B,A]	[**J,K,L**, E,F,G, B,C,D,A]	///J no children: CS=>DE; update SL, NSL; reset CS; update SL
4	K	[K,E,B,A]	[K,L,E,F,G, B,C,D,A]	[**J**]
5	L	[L,E,B,A]	[L,E,F,G, B,C,D,A]	[**K**, J] // remove L, E
6	F	[F, B,A]	[F,G,B,C,D,A]	[**E, L**, K, J]
7	G	[G,B,A]	[G,B,C,D,A]	[**F**, E, L, K, J]
8	M	[M,G,B,A]	[**M,N,H**, G, B,C,D,A]	[F, E, L, K, J]
9	N	[N,G,B,A]	[N,H,G,B,C,D,A]	[**M**, F, E, L, K, J]
10	H	[H,G,B,A]	[H,G,B,C,D,A]	[**N**, M, F, E, L, K, J]
11	O	[O,H,G,B,A]	[**O,P**,H,G,B,C,D,A]	[N, M, F, E, L, K, J]
12	P	[P,H,G,B,A]	[P,H,G,B,C,D,A]	[**O**, N, M, F, E, L, K, J] // remove P,H,G,B
13	C	[C,A]	[C,D,A]	[**B,G,H,P**, O, N, M, F, E, L, K, J]
14	D	[D,A]	[D,A]	[**C**,B,G,H,P, O, N, M, F, E, L, K, J]
15	I	[I,D,A]	[**I**,D,A]	[C,B,G,H,P, O, N, M, F, E, L, K, J]
16	R	[R,I,D,A]	[**R**,I,D,A]	[C,B,G,H,P, O, N, M, F, E, L, K, J]
17	-	[]	[]	[**A,D,I,R**, C,B,G,H,P, O, N, M, F, E, L, K, J]

L5 BFS/DFS

2024年9月20日 20:03

Second Search:

Breath first search: fifo

need large space, but find best, not trapped

Third Search:

Depth first search: L4 Systematically Search

Lifo

记住展开某个节点的子代时候，直接列出所有子代到OPEN列表，然后再考虑最左边的子代的子代

need less space, but not optimal

Iterative DFS: Depth-bound

limit the max depth, not go blind at very beginning

L5 & Summary Topic 2

- **Knowledge Representation**
  - **Symbols**, logic, rules
  - Communicate : can be understood by human and computer
  - Can be processed: structures and computer languages
  - Symbolic AI: Predicate Calculus ; State space Graph
- **Search Strategies**
  - Data-driven vs Goal-driven
  - Graph: Back-tracking
  - BFS: Breadth-First : FIFO queue; shortest path; B^n
  - DFS: Depth-First: FILO stack; dead-end; $B*n$
  - DFS-ID: Depth-Iterative: Depth-bound

L6 Heuristic Search

2024年9月20日 20:03

Uninformed *vs* Informed Search Strategies

Uninformed Strategies use only the information available in the problem definition.

- Systematic generate new states
- Inefficient
- Popular methods:
  - Breadth-first search
  - Depth-First search
  - Iterative Deepening search

Informed Strategies use problem-specific knowledge to guide the search.

- Use an evaluation ***function*** to estimate the 'desirability' of each node
- Usually more efficient
- Popular methods:
  - Hill-Climbing
  - Best-First (Greedy)
  - A*

Summary of those Uninformed Search Algorithms

Search Criteria	BreadthFirst	DepthFirst	IterativeDeepening
Completeness?	Yes	No	Yes
Optimality?	Yes	No	Yes
Time	b^d	b^m	b^d
Space	b^d	bm	bd

Two Key Components of Heuristic Search:
measurement
search alg

1st: Hill climbing

short sighted

2nd: Best first search

can update node measure
所有的叔叔辈，子孙辈统一在open列表里度量最优值

L7 Greedy Astar

2024年9月20日 20:03

Best first:

cost function f(n) = g(n) + h(n)
g is the path cost already determined
h is guessed heuristic further cost
h always underestimated

Greedy best first:

f(n) = h(n)

A star:

f(n) = g(n) + h(n)
combines the strengths of Breadth First Search and Best First.
so to ensure close to the start and find optimal
remind that BFS is only exhausted search, plus heuristic measurement Greedy Search

Properties of Greedy Best-First *vs* A*

Properties of Greedy Best-First – without *g*

Complete ??	No,	can get stuck in loops
Optimal??	No	
Time complexity ??	$O(b^m)$	
Space complexity ??	$O(b^m)$,	keep all nodes in memory

Properties of A* *- h(n) never overestimate cost*

Complete ??	Yes	
Optimal??	Yes	*- if h(n) is both admissible & consistent for graph*
Time complexity ??	$O(b^m)$	
Space complexity ??	$O(b^m)$,	keep all nodes in memory

Criteria	Characteristics
completeness	does it always find a solution if one exists?
optimality	does it always find a least-cost solution?
time complexity	number of nodes generated/expanded
space complexity	maximum number of nodes in memory

Main issue of A* : run out of memory - not practical for many large scale problems

47

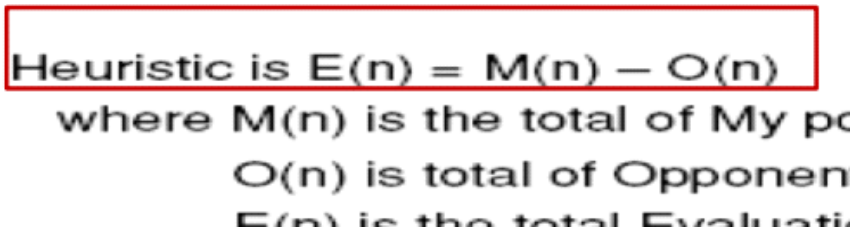
L8 Gaming

2024年9月20日 20:03

Utility function (payoff function): returns a numeric score to quantify the outcome of a game

Minimax procedure on Exhaustively Searchable Graphs - Game tree
if possible, gen full tree and back propagate the 0 or 1 to beginning.
how to calc each step utility? according to win loss or guess a measure.

in tic tac toe:



实际上就是全局倒推每一步该如何决策，或者局部倒推(当树过于庞大时候)

L9 Alpha-Beta Algorithm

2024年9月20日 20:03

Adv MINMAX

alpha for max player, never decrease
beta for min player, never increase

use DFS to temporarily determine the utility

L10 Data Mining

2024年9月20日 20:03

transection data
graph data
sequence data

L11 Association Rule

2024年9月20日 20:04

Basic Concepts

- Let $I = \{ i_1, i_2, \ldots i_d \}$ be the set of d **items** in a dataset, and
- Let $T=\{t_1, t_2, \ldots, t_N\}$ be the set of N **transactions** in a market basket data, where $N=|T|$.
- Each t_j is a subset of I; contains one or a number of items chosen from I.
- A collection of zero or more items is termed as an **itemset**.
- If an itemset contains k items, it is called a ***k*-itemset**, e.g.: {bread, coke, milk}is a 3-itemset.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

- The null (or empty) set is an itemset that does not contain any items.
- An itemset Y is a **superset** of itemset X, or equivalently an itemset X is a **subset** of itemset Y, if all items of X are also items of Y.

More Basic Concepts

- A transaction t_j is said to contain an itemset X if X is a subset of t_j, i.e., $X \subset t_j$. For example, X={Diaper, Milk} and X $\subset t_4$.
- The ***support count*** $\sigma(X)$, of an itemset refers to the **number** of **transactions** that contain itemset X.
- ***Support*** : $\sigma(X)$ / |T|
- An association rule is an implication of the form $X \rightarrow Y$, where

$X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$, e.g., {Diaper, Milk} $\rightarrow$ {Beer},

- The **strength** of an association **rule** can be measured in terms of its *support* and *confidence*.
- ***Support*** specifies how often a rule is applicable in a given dataset.
- ***Confidence*** specifies how frequently items in Y appear in transactions that contain X.

31

对于一个关联对，其support是共现概率，其confidence是条件概率

对于一个itemset，其support是概率

Association Rule Mining (ARM)

1. Frequent Itemset Generation: always expensive
2. Rule Generation

L12 Apriori Principle

2024年9月20日 20:04

The Apriori Principle: If an itemset is frequent, then all its subsets must also be frequent.

Apriori focus on SUPPORT

递进地生成 k-itemset，每一步增加一个
生成时候先生成候选L，其中会有一部分因为上一步的infrequent直接导致本次infrequent
L中开始去原始数据库中计算support，留下合适的成为本次的C

L13 FP-Growth Algorithm

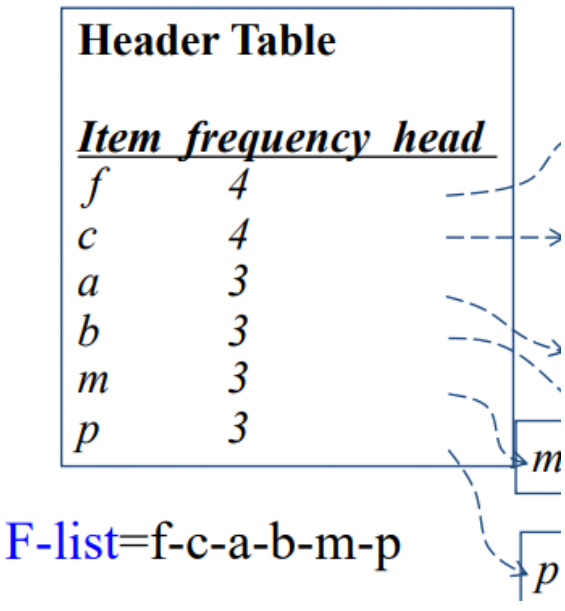
2024年9月20日 20:04

FP-tree (frequent pattern tree)

faster than Apriori!

1. construct whole Tree
2. Use header table, construct conditional pattern base
3. construct conditional FP tree

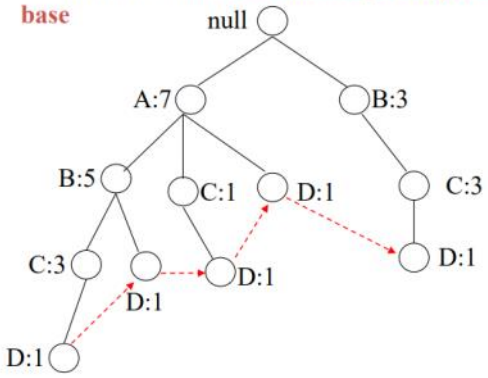
1. **Scan** DB once, find ***frequent*** 1-itemset (single item pattern)
2. **Sort** frequent items in frequency descending order, F-list
3. Scan DB again, **construct** FP-tree



FP-Growth Algorithm

Construct *Conditional Pattern Base*

Starting at the bottom of frequent-item header table in the FP-tree
Traverse the FP-tree by following the link of each frequent item
Accumulate all of transformed prefix paths of that item to form a conditional pattern base



Conditional Pattern base for *D*:
PB = {(A:1,B:1,C:1),
(A:1,B:1),
(A:1,C:1),
(A:1),
(B:1,C:1)}

Recursively apply **FP-Growth** on PatternBase (PB)

Frequent Itemsets found (with *support count* >= 2):
AD, **BD**, **CD**, **ACD**, **BCD**, **ABD** and D as well

53

Association Rule Generation

if $(\sigma(Y)/\sigma(X)) >= minconf$
output rule $X \rightarrow (Y - X)$

from each frequent itemset!!
GEN X->Y-X
so it will always be :
sup(X or Y)/ sup(X)

Association Rules Evaluation

efined as

$$\text{Lift}(X,Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)} = \frac{P(Y \mid X)}{P(Y)}$$

vhere

$$\text{Lift}(X,Y)\begin{cases} = 1, & \text{if } X \text{ and } Y \text{ are independent;} \\ > 1, & \text{if } X \text{ and } Y \text{ are positively correlated;} \\ < 1, & \text{if } X \text{ and } Y \text{ are negatively correlated.} \end{cases}$$

extra

2024年9月21日 10:22

Chen Lihui

2024年11月27日 12:59

State Space Search

2024年11月28日 22:44

Graph/ tuple/ set

- **Initial state:** ({C, F, G, Fm}, { })
- **Goal state:** ({ }, {C, F, G, Fm})
- One state: ({F, G}, {Fm, C})

Bridge Problem

- **Euler's conclusion**: Unless a graph contained either **exactly 0 or 2** nodes of ***odd*** *degree*, a walk over a graph in the manner described by the bridges of Konigsberg problem is impossible

State Space Search

four-tuple [***N, A, S, GD***],

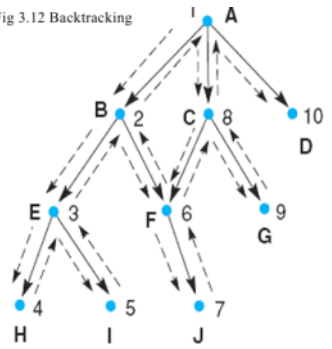
Data-Driven Goal-Driven

Systematical Search:

DFS

BFS

Fig 3.12 Backtracking



Unprocessed

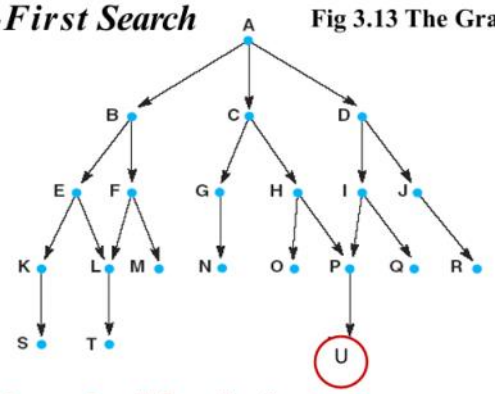
AFTER ITERATION	CS	SL	NSL	DE
0	A	[A]	[A]	[ ]
1	B	[B A]	[B C D A]	[ ]
2	E	[E B A]	[E F B C D A]	[ ]
3	H	[H E B A]	[H I E F B C D A]	[ ]
4	I	[I E B A]	[I E F B C D A]	[H]
5	F	[F B A]	[F B C D A]	[E I H]
6	J	[J F B A]	[J F B C D A]	[E I H]
7	C	[C A]	[C D A]	[B F J E I H]

Example: *Breadth-First Search*

Fig 3.13 The Gra

Trace of ***Breadth-First Search*** on the graph of Fig.3.13 (***U*** is desired goal state)

OPEN: FIFO ; descendant **add from right**
Closed: add from the left



Queue

1. open = [A]; closed = [ ]
2. open = [B,C,D]; closed = [A]
3. open = [C,D,E,F]; closed = [B,A]
4. open = [D,E,F,G,H]; closed = [C,B,A]

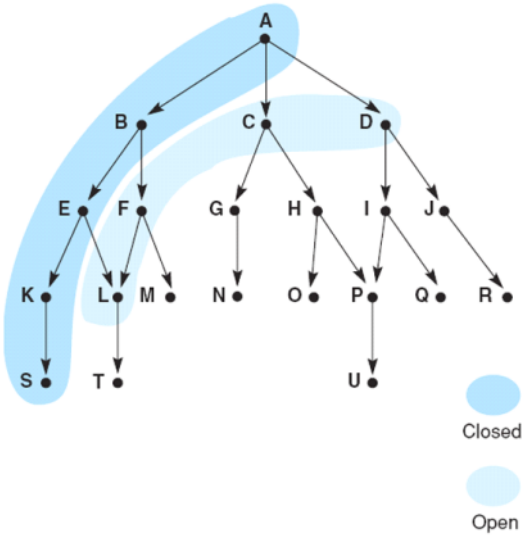
Remove Open(leftmost) => X

generate children of X;
put X on closed;
eliminate children of X on open or closed;

compare to BFS DFS

Advantages : *Breadth-First & Depth-First*

- Advantages of *breadth-first search*:
  - Finds the ***shortest path*** from the start state to the goal
  - Never gets trapped exploring a blind alley
- Advantage of *depth-first* search
  - Depth-first search requires **less memory** since only the nodes on the current path are stored ***B*n***. This contrasts with breadth-first search where all of the tree that has so far been generated B^n



77

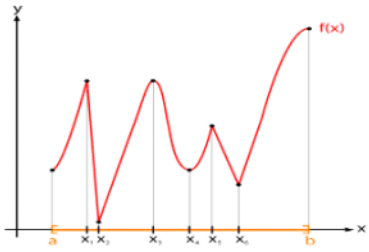
- **Search Strategies**
  - Data-driven vs Goal-driven
  - Graph: Back-tracking
  - BFS: Breadth-First : FIFO queue; shortest path; B^n
  - DFS: Depth-First: FILO stack; dead-end; $B*n$
  - DFS-ID: Depth-Iterative: Depth-bound

Heuristic Search & Gaming

2024年11月29日 0:11

1

Heuristic Search Algorithm- Hill Climbing



Hill climbing: the simplest way

- Named after an eager, but short-sighted mountain climber
  - Hill-climbing strategies expand the current node in the search and evaluate its children
  - The **best child** is selected **for** further **expansion**; neither its siblings nor its parents are **retained**
  - The search halts when it reaches a state that is better than any of its children

Hill Climbing：best first 维护两个列表，open 和close，每次从open展开一项cost最小的

The Best-First

$$f(n) = g(n) + h(n)$$

- $g(n)$: path-cost function = cost from start state to state n
- $h(n)$: heuristic function =Estimated cost from n to a goal state.

The evaluation function $f(n)$ may incorporate various factors such as Expected Remaining Path Cost, Probability of Success, …

we always underestimate the cost in f(n)

Greedy best first, let g(n)=0

A* Search combines the strengths of Breadth First Search and Greedy Best First.

也就是说A*先尽量横向展开所有节点，而且是根据cost有先后的展开

.

Gaming

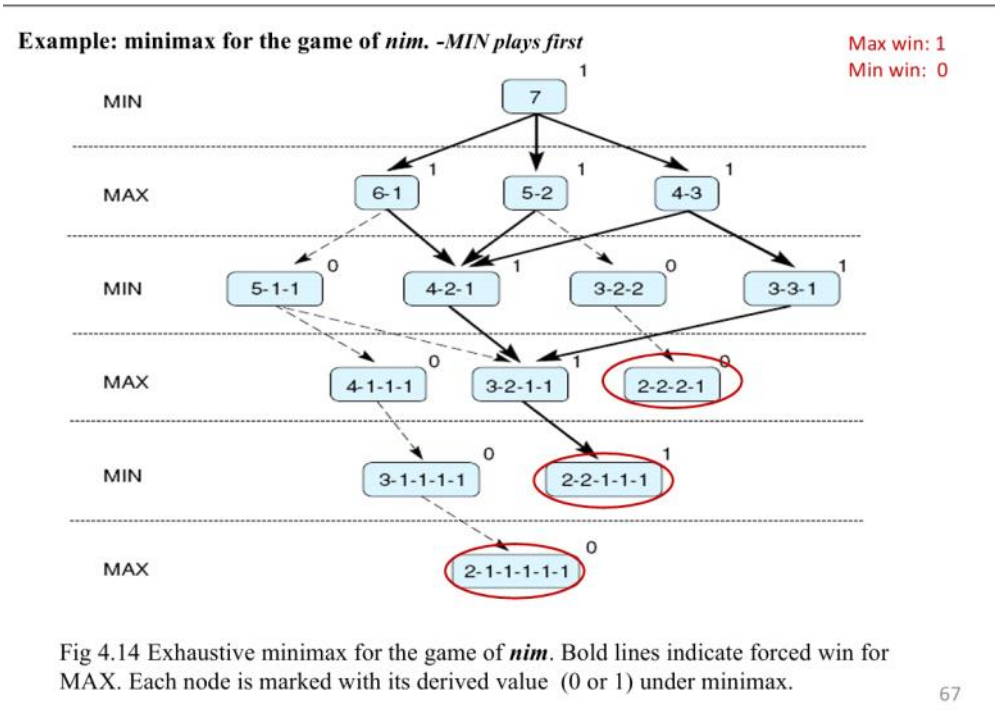
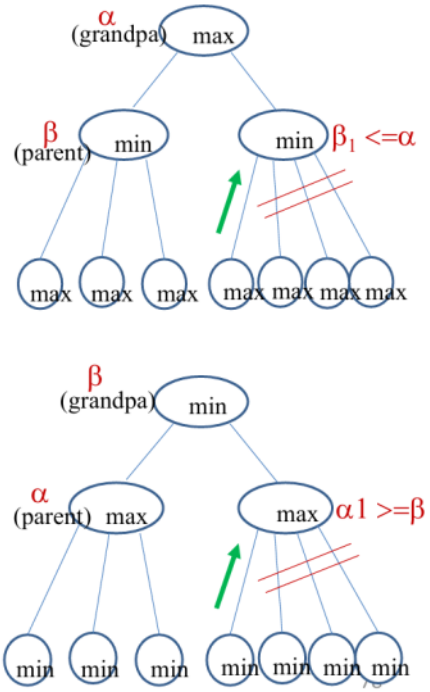


Fig 4.14 Exhaustive minimax for the game of ***nim***. Bold lines indicate forced win for MAX. Each node is marked with its derived value (0 or 1) under minimax.

67

The *alpha-beta* Algorithm Summary

- Two rules for terminating search, based on alpha and beta values, are:
  - Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX ancestors
  - Search can be stopped below any MAX node having an alpha value greater than or equal to the beta value of any of its MIN node ancestors
- Alpha-beta pruning expresses a relation between nodes at **ply *n*** and nodes at **ply *n+2*** under which entire sub-trees rooted at level ***n+1***



Monte Carlo

Association Analysis

2024年11月29日 0:11

关联性分析

Support & Confidence

- The formal definitions of *support* and *confidence* for an association rule $X \rightarrow Y$ in a transaction dataset T:

$$\text{Support}(X) = \sigma(X) / |T| = P(X)$$ *Support* of itemset *X: the Probability of X*

$$\text{Support}(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{|T|} = P(X \cup Y)$$

$$\text{Confidence}(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{P(X \cup Y)}{P(X)} = P(Y \mid X)$$

Example: *X*= { Beer, Milk, Diaper }; **σ(*X*)**= ?

X= { Milk, Diaper }; **σ(*X*)**= ?

Consider the rule {Milk, Diaper} → {Beer}

Support({Milk, Diaper} → {Beer}) = 2/5 = 0.40

Confidence({Milk, Diaper} → {Beer}) = 2/3 = 0.67

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

32

共现概率和条件概率

Frequent Item Set:
Apriori

The *Apriori* Algorithm—Example 0

***minsup* = 2**

C_k: candidate k-itemsets

L_k: frequent k-itemsets.

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan → C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan → C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3'

A B C
A C E
B C E

C_3

Itemset
{B, C, E}

3rd scan → L_3

Itemset	sup
{B, C, E}	2

44

Example: FP-Tree Construction

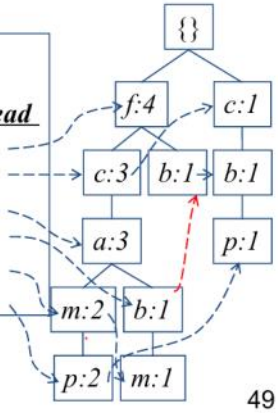
***minsup* = 3**

TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

1. **Scan** DB once, find *frequent* 1-itemset (single item pattern)
2. **Sort** frequent items in frequency descending order, F-list
3. Scan DB again, **construct** FP-tree

Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



F-list=f-c-a-b-m-p

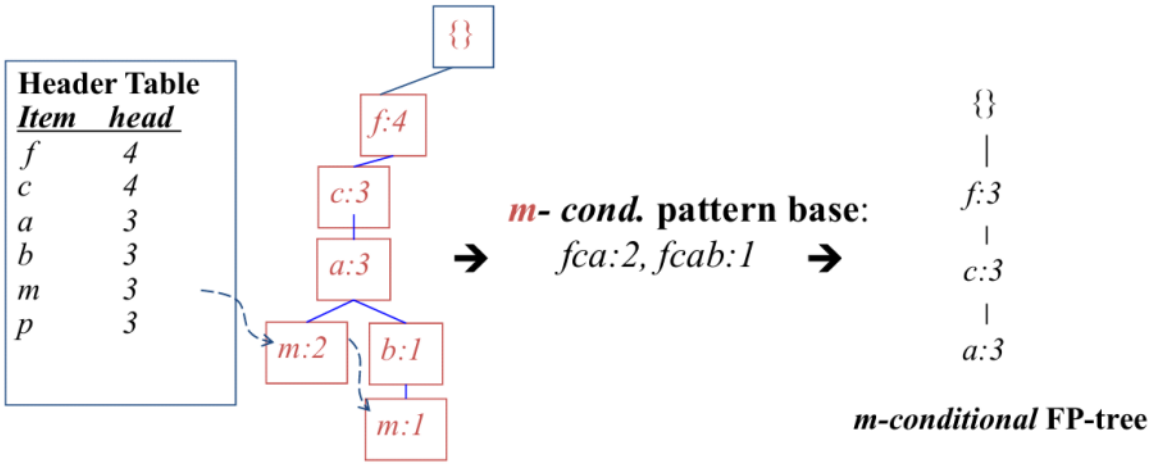
49

Construct *Conditional FP-tree*

For each conditional pattern base

Accumulate the count for each item in the base

Construct the conditional FP-tree for the frequent items of the pattern base



Association Rule Generation

Example1(b). Discussions

TID	Items Bought
1	N, E, W
2	N, O, W
3	W, E
4	O, N, E
5	O, W, N
6	G, O

(a) Minimum support count = $6 \times 0.3 = 1.8 \approx 2$

1-ITEMSET	COUNT
N	4
E	3
W	4
O	4
G	1

2-ITEMSET	COUNT
NE	2
NW	3
NO	3
EW	2
EO	1
WO	2

3-ITEMSET	COUNT
NEW	1
NEO	1
NWO	2

Hence, the frequent itemsets are: N, E, W, O, NE, NW, NO, EW, WO, NWO

(b) A number of association rules have a minimum confidence of 60%. For example:
{NW} → {O} and {NO} → {W}

$$\text{confidence}(\{NW\} \rightarrow \{O\}) = \frac{\sigma(\{NWO\})}{\sigma(\{NW\})} = \frac{2}{3} = 66.7\% > 60\%$$

$$\text{confidence}(\{NO\} \rightarrow \{W\}) = \frac{\sigma(\{NOW\})}{\sigma(\{NO\})} = \frac{2}{3} = 66.7\% > 60\%$$

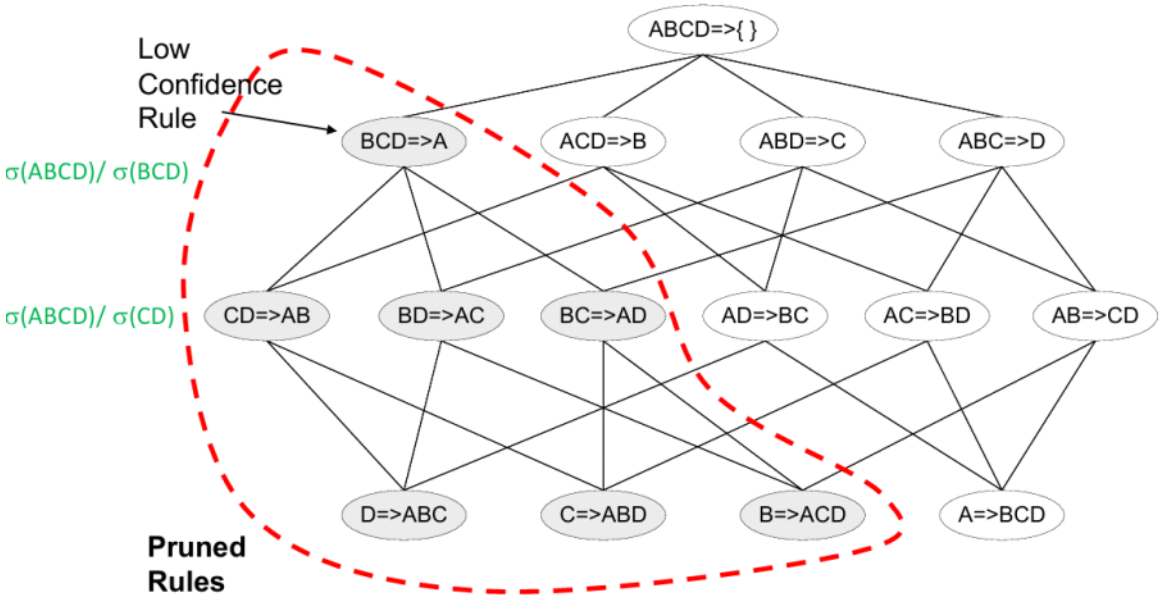
Other possible rules: {N} → {W}, {N} → {O}, {E} → {N}, {E} → {W},
{W} → {N}, {O} → {N}.

60

Confidence-Based Pruning

Y = {A,B,C,D};
X= { B, C, D}

Confidence-Based Pruning: If a rule $X \rightarrow Y\text{-}X$ does not satisfy the confidence threshold, then any rule $X' \rightarrow Y\text{-}X'$, where X' is a subset of X, must not satisfy the confidence threshold as well.



61

[Source: Pang-Ning Tan, et al., "Introduction to Data Mining", Pearson, 2013.]

LIFT

Lift : *interesting* ?

Association Rules Evaluation

- Associate analysis can potentially generate a large number of association rules (patterns).
- Some of these *strong* association rules may still not be interesting.
- Consider the following example of *tea* and *coffee* drinkers:

Association Rule: Tea → Coffee

Confidence=P(Coffee|Tea) = 0.75

but P(Coffee) = 0.8

P(Coffee|$\overline{\text{Tea}}$) = 0.8125 650/800

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

- Although the confidence P(Coffee|Tea) is high, the rule is misleading as the support P(Coffee) and confidence P(Coffee|$\overline{\text{Tea}}$) are even higher.

$$\text{Lift}(X,Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)} = \frac{P(Y \mid X)}{P(Y)}$$

64

Lift: Association Rules Evaluation

To address the problem, an interestingness measure can be used to augment the support-confidence measure for identifying interesting association rules.

Lift is a simple correlation measure between two itemsets X and Y, defined as

$$\text{Lift}(X,Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)} = \frac{P(Y \mid X)}{P(Y)}$$

where

$$\text{Lift}(X,Y) \begin{cases} = 1, & \text{if } X \text{ and } Y \text{ are independent;} \\ > 1, & \text{if } X \text{ and } Y \text{ are positively correlated;} \\ < 1, & \text{if } X \text{ and } Y \text{ are negatively correlated.} \end{cases}$$

For the example of tea-coffee drinkers shown, Lift (Tea, Coffee) = 0.9375, which suggests a slight negative correlation between tea drinkers and coffee drinkers.

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

65

Summary: Topic 4 Association Analysis

- Association rule mining is concerned with finding all **strong** association rules in the form of $X \rightarrow Y$, which satisfies a minimum support threshold ($P(X \cup Y) \geq$ *minsup*) and a minimum confidence threshold ($P(Y \mid X) \geq$ *minconf*).
- Association rule mining first finds all **frequent itemsets** that satisfy the *minsup* threshold, and then extract all **strong** confidence **rules** that satisfy the *minconf* threshold from the frequent itemsets found.
- The *Apriori* algorithm and *FP-Growth* algorithm are efficient methods to mine frequent itemsets from datasets.
- Maximal frequent itemset/Closed frequent itemset
- Not all strong association rules are interesting. Rule (pattern) evaluation measures, such as ***Lift***, a correlation measure, can be used to mine interesting rules.

Tan Yap Peng

2024年11月28日 22:26

Decision Tree

2024年11月28日 22:44

Information Gain

- **Information gain** is used by the ID3^ algorithm as the attribute selection measure .
- The ID3 algorithm minimizes the information (entropy/ uncertainty) in the resulting partitions to achieve the least randomness or "impurity".
- The amount of information in D with m distinct classes can be defined as

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

where m is the number of classes, and p_i is the probability that a sample in D belongs to class c_i, which can be computed as $|C_{i,D}| / |D|$.

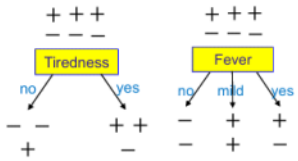
14

^Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81–106

Information Gain

- If attribute A is used to split D into v subsets, $\{D_1, D_2, \ldots, D_v\}$, the resulting information is

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



- Information gain is defined as the difference between the original information (before splitting) and the remaining information (after splitting D by A):

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

- The attribute A which has the highest information gain, Gain(A), is chosen as the splitting attribute.
- The ID3 algorithm is applied recursively on each of the resulting partitions until all the samples are uniquely classified or no further information gain is possible.

15

Information Gain - Example

- Consider the following patient dataset of a healthcare center. The main class of interest is whether a patient is infected with the flu virus (Flu).

PID	Fever	Cough	Sore Throat	Tiredness	Flu
1	no	yes	no	yes	−
2	no	yes	no	no	−
3	mild	yes	no	yes	+
4	yes	mild	no	yes	+
5	yes	no	yes	yes	+
6	yes	no	yes	no	−
7	mild	no	yes	no	+
8	no	mild	no	yes	−
9	no	no	yes	yes	+
10	yes	mild	yes	yes	+
11	no	mild	yes	no	+
12	mild	mild	no	no	+
13	mild	yes	yes	yes	+
14	yes	mild	no	no	−

Disclaimer: Synthetic data

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

$$\begin{aligned} Info_{Fever}(D) &= \frac{5}{14}\times\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+ \frac{4}{14}\times\left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\ &+ \frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.694 \text{ bits} \end{aligned}$$

$$Gain(Fever) = Info(D) - Info_{Fever}(D) = 0.940 - 0.694 = 0.246 \text{ bits}$$

- Similarly, we can obtain $Gain(Cough) = 0.029$ bits, $Gain(Sore\ Throat) = 0.151$ bits and $Gain(Tiredness) = 0.048$ bits.
- Since the "Fever" attribute has the highest information gain, it is selected as the splitting attribute.

16

Gain Ratio

- The attribute with the maximum gain ratio is selected as the splitting attribute.
- The gain ratio of using the "Cough" attribute to split the patient dataset on Slide 16 into three partitions (no, mild, and yes) can be obtained as

$$SplitInfo_{Cough}(D) = -\frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{6}{14}\log_2\left(\frac{6}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) = 1.557 \text{ bits}$$

Since $Gain(Cough) = 0.029$ bits, we have

$$GainRatio_{Cough}(D) = \frac{0.029}{1.557} = 0.019$$

PID	Fever	Cough	Sore Throat	Tiredness	Flu
1	no	yes	no	yes	–
2	no	yes	no	no	–
3	mild	yes	no	yes	+
4	yes	mild	no	yes	+
5	yes	no	yes	yes	+
6	yes	no	yes	no	–
7	mild	no	yes	no	+
8	no	mild	no	yes	–
9	no	no	yes	yes	+
10	yes	mild	yes	yes	+
11	no	mild	yes	no	+
12	mild	mild	no	no	+
13	mild	yes	yes	yes	+
14	yes	mild	no	no	–

19

Gini Index

- Used in CART^, the **Gini index** measures the impurity of D as

$$\text{Gini}(D) = 1 - \sum_{i=1}^{2} p_i^2$$

where p_i is the probability that a sample in D belongs to class C_i and it can be computed as $|C_{i,D}| / |D|$.

- The Gini index considers a binary split for each attribute.
- For example, if a binary split by attribute A resulting in partitions D_1 and D_2, the Gini index is given by

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

^Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees. Wadsworth & Brooks/Cole Advanced Books & Software

20

Evaluating Classifier Performance

- These terms can be summarized in the following confusion matrix:

actual class \ predicted class	Yes	No	Total
Yes	*TP*	*FN*	*P*
No	*FP*	*TN*	*N*
Total	*P'*	*N'*	*P+N*

- The **accuracy** of a classifier is defined as

$$accuracy = \frac{TP + TN}{P + N}$$

- The **error rate** of a classifier can be computed as

$$error\ rate = \frac{FP + FN}{P + N} = 1 - accuracy$$

- The total number of samples is *TP+TN+FP+FN*, or *P+N*, or *P'+N'*.

28

Evaluating Classifier Performance

- The **sensitivity** (true positive) and **specificity** (true negative) measure how well a classifier can correctly recognize the positive samples and negative samples, respectively.

$$sensitivity = \frac{TP}{P} \qquad specificity = \frac{TN}{N}$$

- The **accuracy** is a function of sensitivity and specificity, given as

$$accuracy = sensitivity \times \left(\frac{P}{P+N}\right) + specificity \times \left(\frac{N}{P+N}\right)$$

29

Evaluating Classifier Performance

- **Precision** is the percentage of samples labeled as positive that are actually positive, while **recall** is the percentage of positive samples that are correctly labeled as positive.

$$precision = \frac{TP}{TP+FP} = \frac{TP}{P'}$$

$$recall = \frac{TP}{TP+FN} = \frac{TP}{P}$$

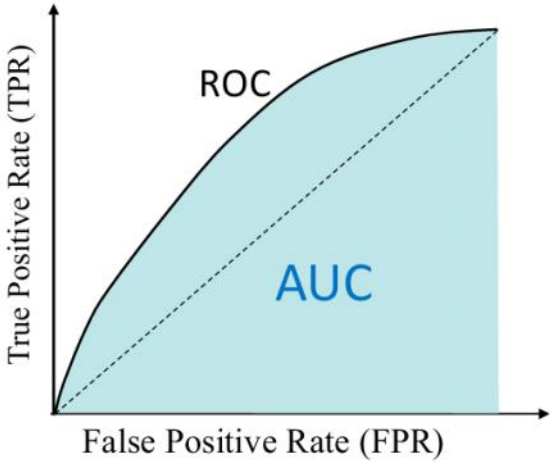
- For convenience, precision and recall can also be combined into a single measure, e.g., the ***F* measure** is defined as

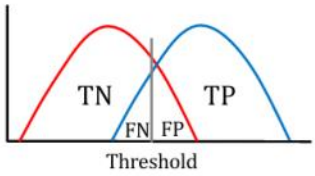
$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Evaluating Classifier Performance

- Area Under the Curve (AUC) of the Receiver Operating Characteristics (ROC) measures the classification performance at various threshold settings.

$$TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$$





$$FPR = \frac{FP}{TN+FP} = \frac{FP}{N}$$

Evaluating Classifier Performance

- Consider the following confusion matrix for a medical dataset where the class values are ***yes*** and ***no*** for the flu virus.

actual class \ predicted class	Yes	No	Total
Yes	100	200	300
No	150	9550	9700
Total	250	9750	10000

	Yes	No	Total
Yes	*TP*	*FN*	*P*
No	*FP*	*TN*	*N*
Total	*P'*	*N'*	*P+N*

The following performance measures can be obtained:

$$sensitivity = \frac{100}{300} = 33.33\% \qquad specificity = \frac{9550}{9700} = 98.45\%$$

$$accuracy = \frac{9650}{10000} = 96.50\% \qquad precision = \frac{100}{250} = 40\%$$

$$recall = \frac{100}{300} = 33.33\% \qquad F = \frac{2\times0.4\times0.3333}{0.4+0.3333} = 36.36\%$$

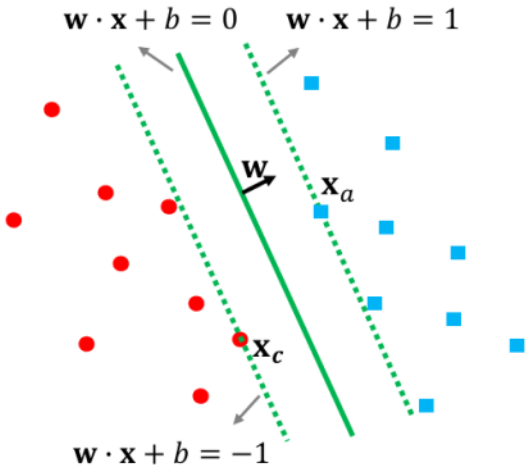
32

SVM: Linear Decision Boundary

- If $\mathbf{x}_a$ is located above the decision boundary and $\mathbf{x}_c$ is below, we have

$$\mathbf{w} \cdot \mathbf{x}_a + b > 0$$

$$\mathbf{w} \cdot \mathbf{x}_c + b < 0$$



- Label all squares as class $+1$ and all the circles as class -1; the label of any test example $\mathbf{z}$ can be predicted as:

$$y = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b > 0; \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b < 0. \end{cases}$$

16

SVM: Margin

- Let the distance of the two hyperplanes, which are parallel to the decision boundary and cross the two closest samples, be

$$\mathbf{w} \cdot \mathbf{x}_a + b = k > 0$$
$$\mathbf{w} \cdot \mathbf{x}_c + b = k' < 0$$

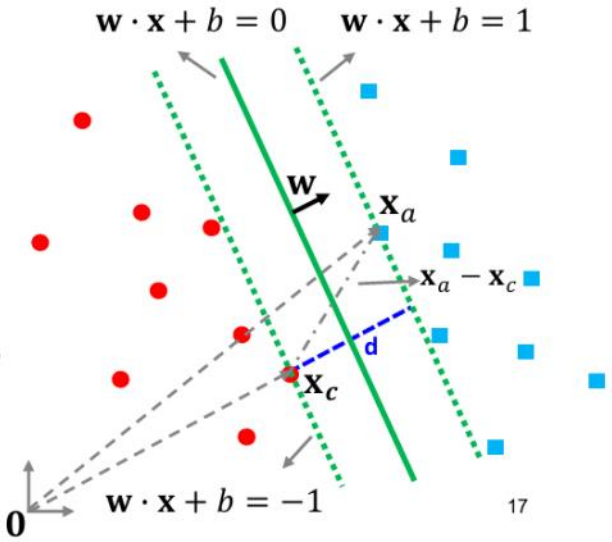
- We can rescale the parameters $\mathbf{w}$ and b to express the two hyperplanes as:

$$\mathbf{w} \cdot \mathbf{x}_a + b = 1$$
$$\mathbf{w} \cdot \mathbf{x}_c + b = -1$$

- The margin d can be obtained as

$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_c) = 2$$
$$\|\mathbf{w}\| \times d = 2 \implies d = \frac{2}{\|\mathbf{w}\|}$$

- Samples $\mathbf{x}_a$ and $\mathbf{x}_c$ are known as support vectors.



Linear SVM Model

- SVM learns the parameters $\mathbf{w}$ and b of the decision boundary to satisfy all training samples, as follows:

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x_i}+b &\geq 1 \text{ if } y_i=1,\\ \mathbf{w}\cdot\mathbf{x_i}+b &\leq -1 \text{ if } y_i=-1. \end{aligned} \quad \Rightarrow \quad y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geq 1, \quad i=1,2,\ldots,N.$$

- Besides, the margin d of the decision boundary must be maximum:

$$\max d = \frac{2}{\|\mathbf{w}\|} \quad \Rightarrow \quad \min \frac{\|\mathbf{w}\|^2}{2}$$

- The SVM can be learned by solving the following constrained optimization problem with the standard Lagrange multiplier method.

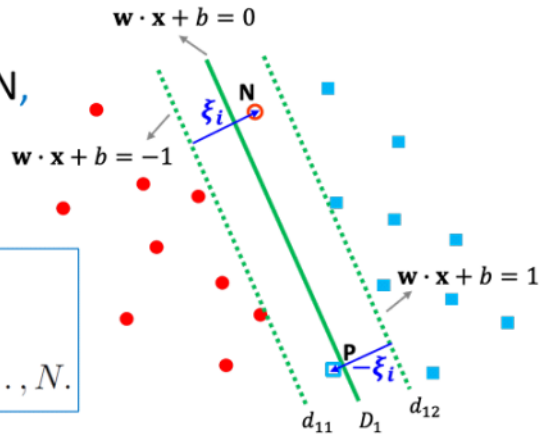
$$\begin{aligned} &\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}\\ &\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geq 1, \quad i=1,2,\ldots,N. \end{aligned}$$

18

SVM: Non-separable Cases

- **Soft margin approach**: To trade-off between the margin's width and the training errors, learn a decision boundary that allows small training errors.
- With the new training samples P and N, the decision boundary D_1 no longer satisfies the original constraints:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}\\ &\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geq 1, \quad i=1,2,\ldots,N. \end{aligned}$$



- The inequality constraints can be relaxed to accommodate the deviation by using a **slack variable $\xi_i > 0$**:

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x_i}+b &\geq 1-\xi_i \text{ if } y_i=1,\\ \mathbf{w}\cdot\mathbf{x_i}+b &\leq -1+\xi_i \text{ if } y_i=-1, \end{aligned} \quad \Rightarrow \quad y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1-\xi_i$$

20

Neural Networks

2024年11月30日 18:30

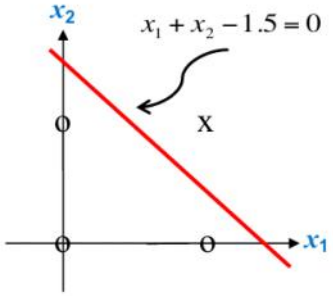
Perceptrons - Examples

Examples of linearly separable classes

- Logical AND function

$$f(X) = \begin{cases} 0, & \text{if } X \le 0 \\ 1, & \text{if } X > 0 \end{cases}$$

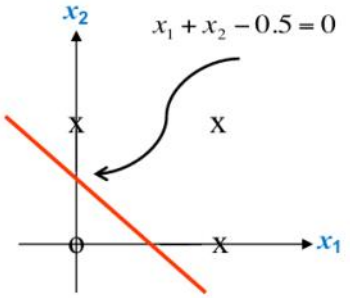
x_1	x_2	y	$y = \sigma(w_1x_1 + w_2x_2 + b)$	
0	0	0	$w_1 0 + w_2 0 + b \le 0$	$b \le 0$
0	1	0	$w_1 0 + w_2 1 + b \le 0$	$w_2 \le -b$
1	0	0	$w_1 1 + w_2 0 + b \le 0$	$w_1 \le -b$
1	1	1	$w_1 1 + w_2 1 + b > 0$	$w_1 + w_2 > -b$



o: Class 0 (y = 0)
x: Class 1 (y = 1)

- Logical OR function

x_1	x_2	y	$y = \sigma(w_1x_1 + w_2x_2 + b)$	
0	0	0	$w_1 0 + w_2 0 + b \le 0$	$\theta \le 0$
0	1	1	$w_1 0 + w_2 1 + b > 0$	$w_2 > -b$
1	0	1	$w_1 1 + w_2 0 + b > 0$	$w_1 > -b$
1	1	1	$w_1 1 + w_2 1 + b > 0$	$w_1 + w_2 > -b$



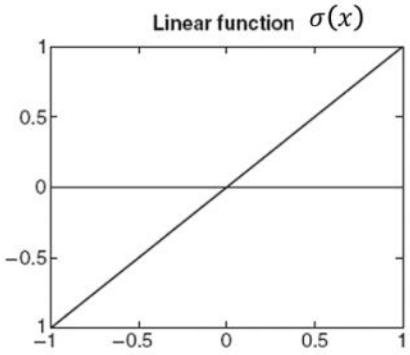
o: Class 0 (y = 0)
x: Class 1 (y = 1)

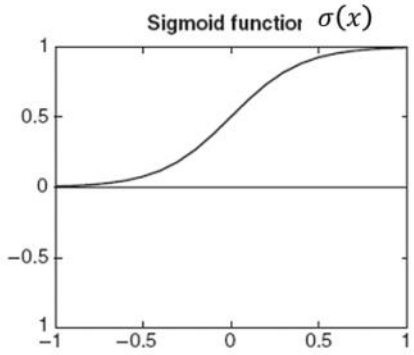
6

Activation Functions

- Common activation functions:

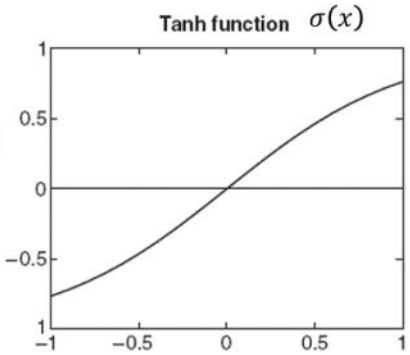
$\sigma(x) = x$

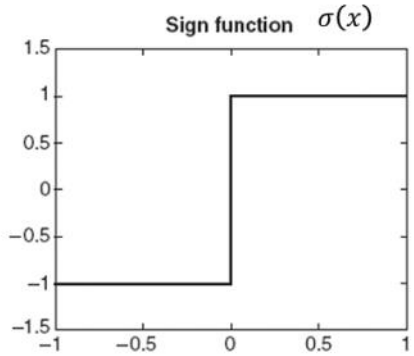




$\sigma(x) = \frac{1}{1+e^{-ax}}$

$\sigma(x) = \tanh(\gamma x)$
$= \frac{e^{2\gamma x}-1}{e^{2\gamma x}+1}$





$\sigma(x) = \text{sign}(x)$
$= \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$

11

The Backpropagation Algorithm

- Consider using backpropagation with gradient descent to minimize the sum of squared errors (a loss function):

$$E = \frac{1}{2}\sum_{k=1}^{c}(t_k - o_k)^2 = \frac{1}{2}\|\boldsymbol{t}-\boldsymbol{o}\|^2 \qquad \Delta w_{jk} = -\eta_w \frac{\partial E}{\partial w_{jk}} \qquad \Delta b_j = -\eta_b \frac{\partial E}{\partial b_j}$$

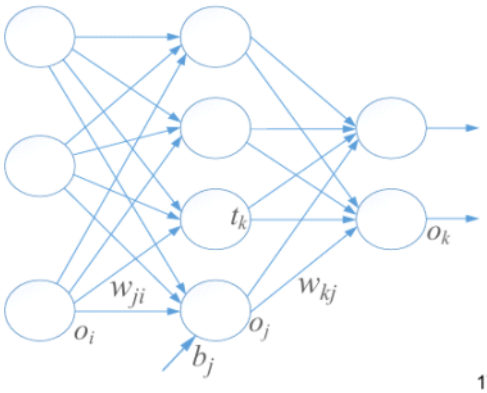
where $\boldsymbol{t}$ and $\boldsymbol{o}$ are the target and network output vectors.

- The key steps of the backprogagation algorithm are:

1) Initialize all the weights and biases.
2) Propagate the inputs forward and compute the net input net_j and output o_j for each output and hidden unit.

$$net_j = \sum_i w_{ji} o_i + b_j$$
$$o_j = \sigma(net_j)$$

where $\sigma(x)$ is the activation function.



17

The Backpropagation Algorithm

3) Propagate the error backward by updating the error for each output and hidden unit.

$$\delta_k = \sigma'(net_k)(t_k - o_k) \quad \text{for output units}$$

$$\delta_j = \sigma'(net_j)\sum_k \delta_k w_{kj} \quad \text{for hidden units}$$

Note: If $\sigma(x) = 1/(1+e^{-x})$ is the activation function, we have

$$\delta_k = o_k(1-o_k)(t_k - o_k) \quad \text{for output units}$$

$$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj} \quad \text{for hidden units}$$

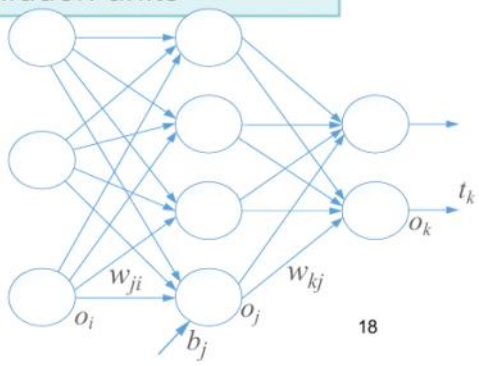
4) Update the weights and biases, as follows:

$$w_{kj} = w_{kj} + \Delta w_{kj} \quad \text{where} \quad \Delta w_{kj} = \eta_w \delta_k o_j$$

$$w_{ji} = w_{ji} + \Delta w_{ji} \quad \text{where} \quad \Delta w_{ji} = \eta_w \delta_j o_i$$

$$b_j = b_j + \Delta b_j \quad \text{where} \quad \Delta b_j = \eta_b \delta_j$$

where η_w and η_b are learning rates.



18

The Backpropagation Algorithm

- Steps (2) to (4) are repeated until a termination condition is met, e.g.,
  - The value of the error function is below a predefined threshold.
  - All updates are smaller than some predefined threshold(s).
  - A predetermined total number of updates has been reached.

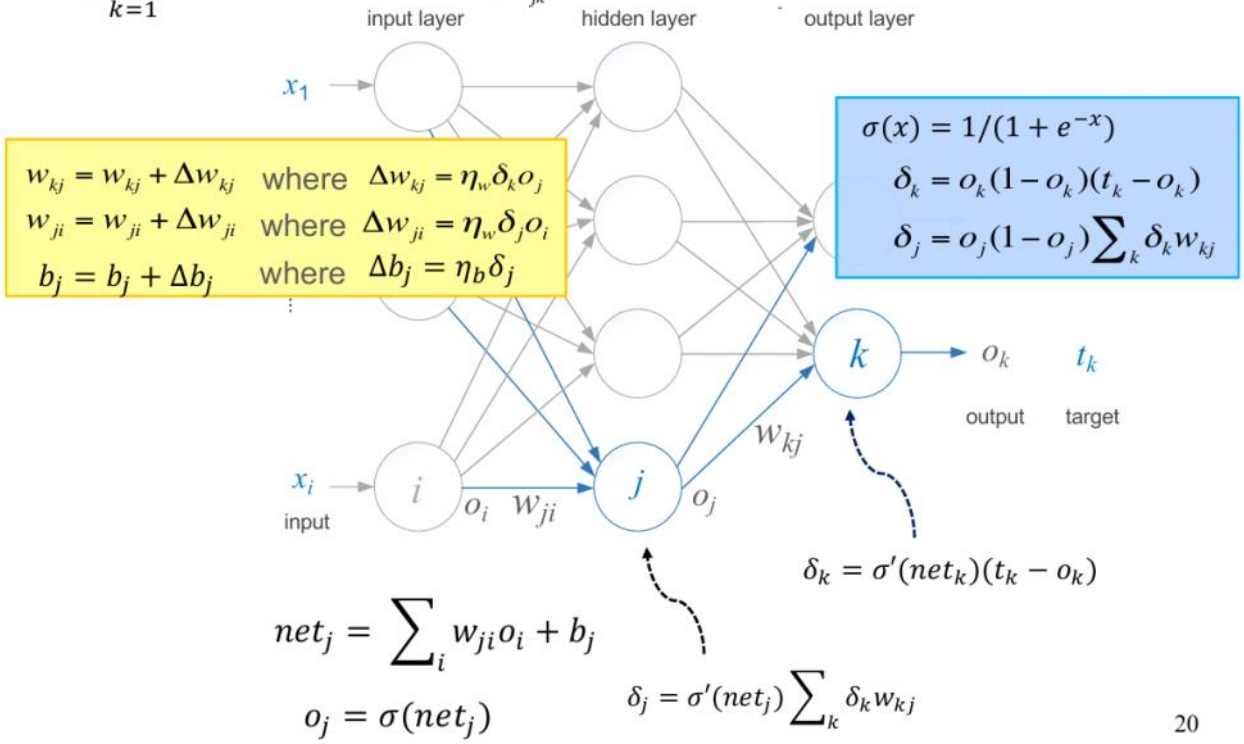
- The weights and biases can be updated in different ways:
  - Case updating: Updates are made after each data sample is presented.
  - Epoch updating: Increments are accumulated and the updates are made only after all samples in the training set have been presented.
  - Stochastic gradient descent: A small number of data samples are randomly selected to perform the update. Repeat this process until every sample in the training set has been used.

19

The Backpropagation Algorithm

[Optional]

$$E = \frac{1}{2}\sum_{k=1}^{c}(t_k - o_k)^2 \qquad \Delta w_{jk} = -\eta_w \frac{\partial E}{\partial w_{jk}} \qquad \Delta b_j = -\eta_b \frac{\partial E}{\partial b_j}$$



$$w_{kj} = w_{kj} + \Delta w_{kj} \quad \text{where} \quad \Delta w_{kj} = \eta_w \delta_k o_j$$
$$w_{ji} = w_{ji} + \Delta w_{ji} \quad \text{where} \quad \Delta w_{ji} = \eta_w \delta_j o_i$$
$$b_j = b_j + \Delta b_j \quad \text{where} \quad \Delta b_j = \eta_b \delta_j$$

$$\sigma(x) = 1/(1+e^{-x})$$
$$\delta_k = o_k(1-o_k)(t_k - o_k)$$
$$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj}$$

$$\delta_k = \sigma'(net_k)(t_k - o_k)$$

$$net_j = \sum_i w_{ji} o_i + b_j$$

$$o_j = \sigma(net_j)$$

$$\delta_j = \sigma'(net_j)\sum_k \delta_k w_{kj}$$

The Backpropagation Algorithm

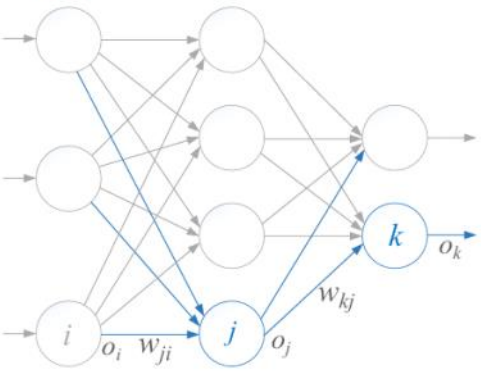
[Optional]

Given $net_k = \sum_j w_{kj} o_j + b_k \qquad o_k = \sigma(net_k)$

$net_j = \sum_i w_{ji} o_i + b_j \qquad o_j = \sigma(net_j)$

$E = \frac{1}{2}\sum_{k=1}^{c}(t_k - o_k)^2 \qquad w_{kj} = w_{kj} + \Delta w_{kj}$

$\Delta w_{kj} = -\eta_w \frac{\partial E}{\partial w_{kj}}$



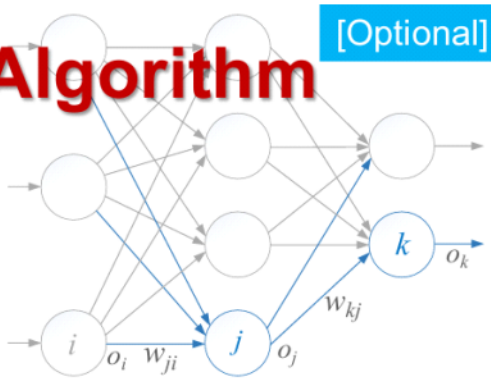
Consider output unit k and weight w_{kj}

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} \qquad \text{[applying chain rule]}$$
$$= \frac{\partial E}{\partial o_k} \cdot \frac{\partial \sigma(net_k)}{\partial net_k} \cdot \frac{\partial\left(\sum_j w_{kj} o_j + b_k\right)}{\partial w_{kj}}$$
$$= -(t_k - o_k) \cdot \sigma'(net_k) \cdot o_j$$
$$= -\delta_k \cdot o_j$$

where $\delta_k = \sigma'(net_k) \cdot (t_k - o_k)$ and hence $\Delta w_{kj} = \eta_w \delta_k o_j$

The Backpropagation Algorithm

[Optional]



Given $w_{ji} = w_{ji} + \Delta w_{ji}$ $\Delta w_{ji} = -\eta_w \frac{\partial E}{\partial w_{ji}}$

Consider hidden unit j and weight w_{ji}

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} \\ &= \frac{\partial E}{\partial o_j} \cdot \frac{\partial \sigma(net_j)}{\partial net_j} \cdot \frac{\partial \left(\sum_i w_{ji} o_i + b_j\right)}{\partial w_{ji}} \\ &= -\left(\sum_k \delta_k w_{kj}\right) \cdot \sigma'(net_j) \cdot o_i \\ &= -\delta_j \cdot o_i\end{aligned}$$

Note:

$$\begin{aligned}\frac{\partial E}{\partial o_j} &= \frac{\partial \left[\frac{1}{2}\sum_k (t_k - o_k)^2\right]}{\partial o_j} = \frac{1}{2}\sum_k \frac{\partial (t_k - o_k)^2}{\partial o_k} \cdot \frac{\partial o_k}{\partial o_j} \\ &= -\sum_k (t_k - o_k) \cdot \frac{\partial o_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j} \\ &= -\sum_k (t_k - o_k) \cdot \frac{\partial \sigma(net_k)}{\partial net_k} \cdot \frac{\partial \left(\sum_j w_{kj} o_j + b_k\right)}{\partial o_j} \\ &= -\sum_k (t_k - o_k) \cdot \sigma'(net_k) \cdot w_{kj} \\ &= -\sum_k \delta_k w_{kj}\end{aligned}$$

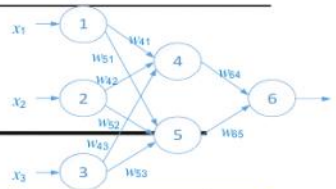
where $\delta_j = \sigma'(net_j) \cdot \sum_k \delta_k w_{kj}$ and hence $\Delta w_{ji} = \eta_w \delta_j o_i$

22

Backpropagation - Example

- Given the training sample x=(1,0,1), the net input and output of each unit can be computed as

Unit j	Net input net_j	$net_j = \sum_i w_{ji} o_i + b_j$	Output o_j	$o_j = \sigma(net_j)$
4	0.2+0-0.5-0.4 = -0.7		$1/(1+e^{0.7})$=0.332	
5	-0.3+0+0.2+0.2=0.1		$1/(1+e^{-0.1})$=0.525	
6	(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105		$1/(1+e^{0.105})$=0.474	



- The error of each unit can be obtained as

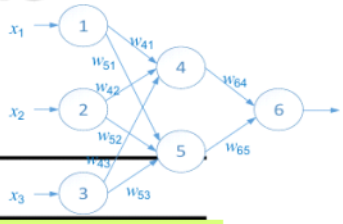
Unit j	Err $_j$	
4	(0.332)(1-0.332)(0.1311)(-0.3) = -0.0087	$\delta_j = \sigma'(net_j) \cdot \sum_k \delta_k w_{kj}$
5	(0.525)(1-0.525)(0.1311)(-0.2) = -0.0065	
6	(0.474)(1-0.474)(1-0.474)= 0.1311	$\delta_k = \sigma'(net_k) \cdot (t_k - o_k)$

24

[Source: Jiawei Han, et al. "Data Mining: Concepts and Techniques", 3rd Edition, Morgan Kaufmann, 2011.]

Backpropagation - Example

- The weights and biases can be updated as



Weight/Bias	New Value	
w_{64}	-0.3+(0.9)(0.1311)(0.332) = -0.261	$\Delta w_{kj} = \eta_w \delta_k o_j$
w_{65}	-0.2+(0.9)(0.1311)(0.525) = -0.138	
w_{41}	0.2+(0.9)(-0.0087)(1)= 0.192	$\Delta w_{ji} = \eta_w \delta_j o_i$
w_{51}	-0.3+(0.9)(-0.0065)(1)= -0.306	
w_{42}	0.4+(0.9)(-0.0087)(0)= 0.4	
w_{52}	0.1+(0.9)(-0.0065)(0)= 0.1	
w_{43}	-0.5+(0.9)(-0.0087)(1) = -0.508	
w_{53}	0.2+(0.9)(-0.0065)(1)= 0.194	
b_6	0.1+(0.9)(0.1311) = 0.218	$\Delta b_j = \eta_b \delta_j$
b_5	0.2+(0.9)(-0.0065) = 0.194	
b_4	-0.4+(0.9)(-0.0087) = -0.408	

25

[Source: Jiawei Han, et al. "Data Mining: Concepts and Techniques", 3rd Edition, Morgan Kaufmann, 2011.]

Wen Bihan

2024年11月28日 22:26

Clustering Regression

2024年11月30日 18:30

K means/ HAC

K-Means vs. HAC

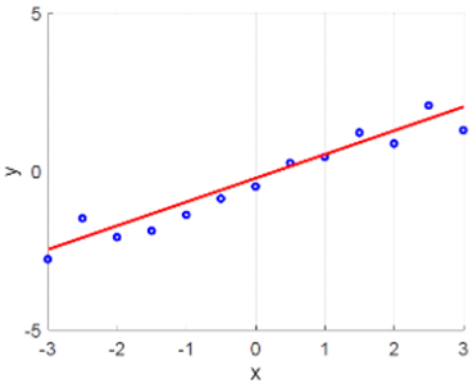
- K-Means
  - ✓ Simple and cheap algorithm
  - ✗ Results are sensitive to the initialization
  - ✗ Number of clusters needs to be pre-defined
- HAC
  - ✓ Deterministic algorithm, i.e., not randomness.
  - ✓ Show us a range of clustering results with different choices of K.
  - ✗ More memory- and computationally-intensive than K-Means

Linear Regression - Derivation

- As we are minimizing the *mean squared error*, we call this solution as the **least square (LS)** estimator:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Some times linear is not good enough…

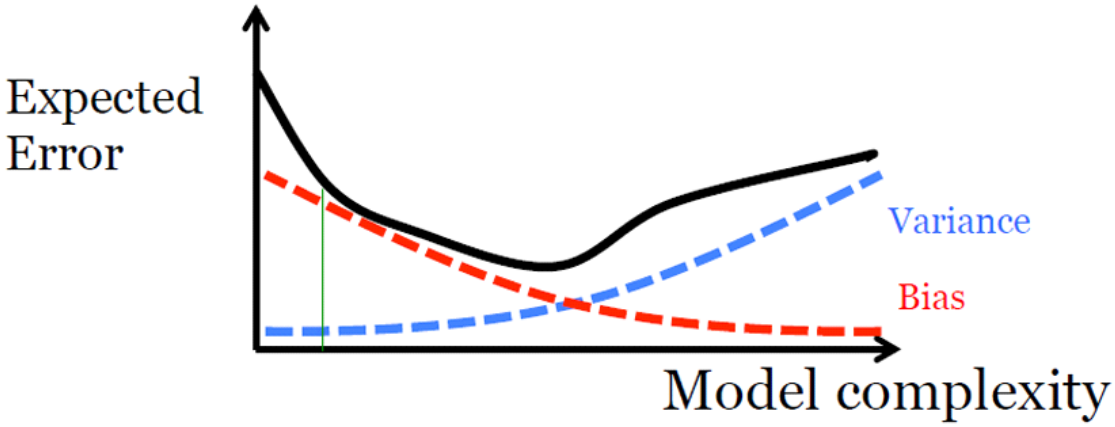


Regularization

2024年11月30日 18:30

Bias and Variance

- Training a classifier $f_\theta(x)$



- **Expected error** of a classifier ≈ **bias** + **variance** (+noise)

PCA Bayesian

2024年11月30日 19:34

Principal Component Analysis

- Two ways to interpret the "**Best**":
  1. The mean square error **(MSE)** of the projected data is **minimized**.
  2. The **variance** of the projected data is **maximized**.
- These two objectives are **achieved simultaneously** by applying PCA for dimensionality reduction.
- Why and How?

19

Bayes' Theorem

- Now, we can predict $P(h|D)$:

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

- Some terminologies we are going to use:
  - **Prior probability** $P(h)$: prior knowledge of h **before** observing D.
  - **Posterior** $P(h|D)$: the probability of h **after** we have observed D.
  - **Likelihood** $P(D|h)$: Likelihood of observing D given h.