Knowledge Representation:
Capturing knowledge in a way suitable for computer manipulation.
– Predicate Calculus/Neural Network; – Graph / Tree

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.

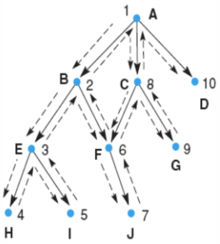
Graph Terminologies:
– Node/Arch/Path/Tree; – Directed/Rooted Graphs; – Parent, Siblings/Ancestor/Descendant

State Space Approach Examples:
– Tic-Tac-Toe/8-puzzle; – TSP: The number of possible ways to visit N cities, (N-1)!

Backtracking:
– Depth-first search for CSPs; – Basic uninformed search for CSPs
Notations:
– 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.

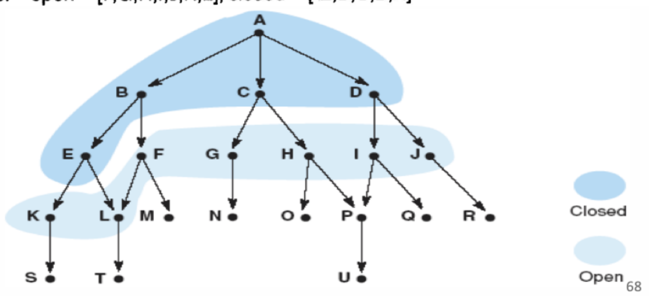
Suppose *G* is the Goal State **Backtracking – example**

Iteration	CS –size1	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
	E	E B A	E F B C D A	I 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
	F	F B A	F B C D A	J E I H
	B	B A	B C D A	F J E I H
7	C	C A	C D A	B F J E I H
8	G	G C A	G C D A	B F J E I H

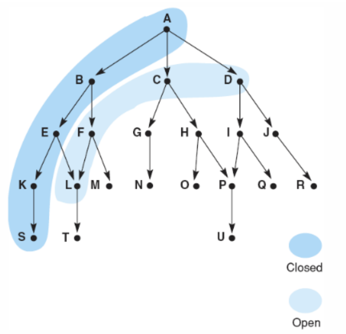


Breadth-first:
– open - states that have been generated but whose children have not been examined. Right in, left out; first-in-first-out. (FIFO)
– closed - states that have already been examined. Add from the left.
– Memory used: B^n

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]
5. open = [E,F,G,H,I,J]; closed = [D,C,B,A]
6. open = [F,G,H,I,J,K,L]; closed = [E,D,C,B,A]



Depth-first:
– open is maintained as a stack, or last-in-first- out (LIFO) structure. Open is similar to NSL in backtrack
– closed- states that have already been examined. An union of DE and SL in backtrack
– Memory used: $B * n$



Trace of depth_first_search on figure 3.13

1. open = [A]; closed = [ ]
2. open = [B,C,D]; closed = [A]
3. open = [E,F,C,D]; closed = [B,A]
4. open = [K,L,F,C,D]; closed = [E,B,A]
5. open = [S,L,F,C,D]; closed = [K,E,B,A]
6. open = [L,F,C,D]; closed = [S,K,E,B,A]
7. open = [T,F,C,D]; closed = [L,S,K,E,B,A]
8. open = [F,C,D]; closed = [T,L,S,K,E,B,A]
9. open = [M,C,D], L is already on closed; closed = [F,T,L,S,K,E,B,A]
10. open = [C,D]; closed = [M,F,T,L,S,K,E,B,A]

and so on until either U is discovered or open = [ ]

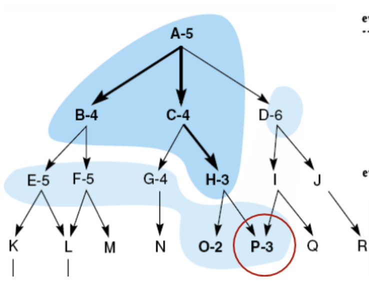
Depth-First with Iterative Deepening:
Depth bound from 1, and increase by one each time.

Uninformed:
BFS: b^d, b^d; DFS: b^m, $b * m$; IDS: b^d, $b * d$.
b- maximum branching factor of the search tree; d- depth of the least-cost solution; m- maximum depth of the state space (may be unlimited)

Informed: Hill-Climbing, Best-First(Greedy), A*

Heuristic Search:
– Hill-Climbing
– Best-First-Search:

Fig 4.5 Heuristic search of a hypothetical state space with open and closed states highlighted.



The Trace of best-first-search on Figure 4.4:

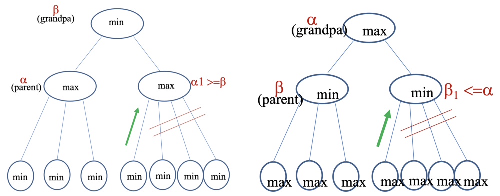
Initialize: open = [A5], closed = [ ]

evaluate	open	closed
A5	[B4,C4,D6]	[A5]
B4	[C4,E5,F5,D6]	[B4,A5]
C4	[H3,G4,E5,F5,D6]	[C4,B4,A5]
H3	[O2,P3,G4,E5,F5,D6]	[H3,C4,B4,A5]
O2	[P3,G4,E5,F5,D6]	[O2,H3,C4,B4,A5]

evaluate P3; the solution is found!

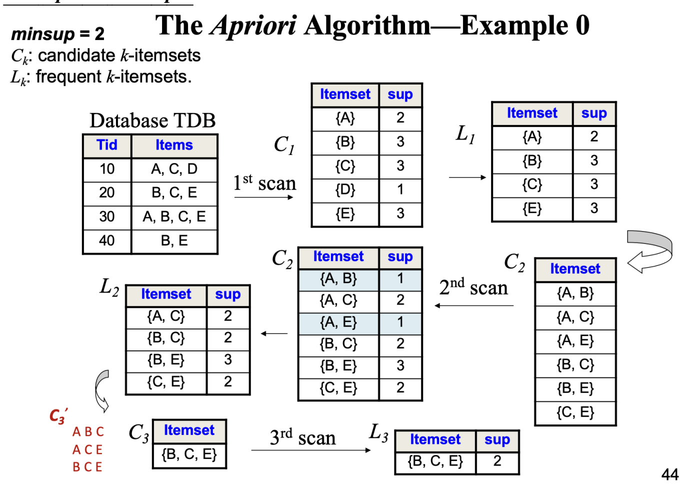
- Evaluation function f(n)=g(n)+h(n):
g(n) = cost so far to reach n; h(n) = estimated cost to goal from n; f(n) = estimated total cost of path through n to goal.
– When g(n)=0, Greedy Best-First; – A* search is optimal, when h(n) is admissible. h(n) is always under-estimated/same as the actual cost from n to a goal.

Minimax:
– ALPHA-BETA pruning: Directly prune the whole right node.



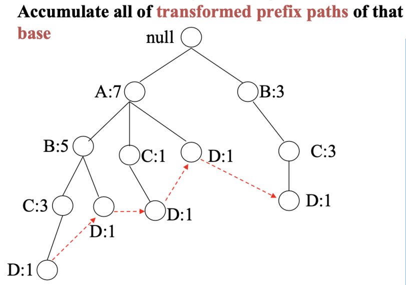
Association rule:
An *association rule* is an implication of the form $X \to Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$, e.g., $\{\text{Diaper, Milk}\} \to \{\text{Beer}\}$,
$\text{Support}(X) = \frac{\sigma(X)}{|T|} = P(X)$ *Support of itemset* X *: the Probability of* X
$\text{Support}(X \to Y) = \frac{\sigma(X \cup Y)}{|T|} = P(X \cup Y)$
$\text{Confidence}(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{P(X \cup Y)}{P(X)} = P(Y \mid X)$ There are a total of $3^d - 2^{d+1} + 1$ possible rules for a dataset containing d items. $2^d - 1$ item sets.

The Apriori Principle:



The FP-Growth Algorithm:

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

Association Rule Generation:
if $\frac{\sigma(Y)}{\sigma(X)} \geq minconf$ $X \subset Y$, $X \to Y - X$. If $|Y| = k$, then there are $2^k - 2$ candidate association rules (ignoring: $Y \to \emptyset$ and $\emptyset \to Y$).
Lift is a simple correlation measure between two item sets X and Y, defined as
$\text{Lift}(X,Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)} = \frac{P(Y|X)}{P(Y)}$

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

Information Gain:
The amount of information in D with m distinct classes can be defined as:
$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$
If attribute A is used to split D into v subsets, $\{D_1, D_2, \ldots, D_v\}$, the resulting information is

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

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

$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$

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

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

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

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

$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right)$

$GainRatio_A(D) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}$

Gini Index:

$\text{Gini}(D) = 1 - \sum_{i=1}^{2} p_i^2 \quad \text{Gini}_A(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$

$\Delta\text{Gini}(A) = \text{Gini}(D) - \text{Gini}_A(D)$

Evaluating Classifier Performance:

predicted class

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

Accuracy $= \frac{TP+TN}{P+N}$ Error rate $= \frac{FP+FN}{P+N} = 1 - \text{Accuracy}$

Sensitivity $= \frac{TP}{P}$ Specificity $= \frac{TN}{N}$

Accuracy $=$ Sensitivity $\times \left(\frac{P}{P+N}\right) +$ Specificity $\times \left(\frac{N}{P+N}\right)$

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

Recall $= \frac{TP}{TP+FN} = \frac{TP}{P} = TPR \quad \frac{FP}{N} = FPR$

$F = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

Dissimilarity and Similarity Measures:

1. Minkowski distance: $d(\mathbf{x},\mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r\right)^{1/r}$
2. Manhattan distance ($r=1$): $d(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{n} |x_k - y_k|$
3. Euclidean distance ($r=2$): $d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$
4. Cosine similarity: $\cos(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$
5. Infinity (Sup) Distance: $d(\mathbf{x},\mathbf{y}) = \max_{1\le j\le d} |x_j - y_j|$

SVM:

$y_i = \text{sign}(\mathbf{w}\cdot\mathbf{x}_i + b). \quad d = \frac{2}{\|\mathbf{w}\|}.$

$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$ subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \ge 1,\ i = 1,2,\ldots,N.$

– Dual optimization problem

$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{N} \lambda_i y_i = 0.$

$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 = 0.$

Neural Network:

– Activation Functions:

Linear: $\sigma(x) = x$ Sigmoid: $\sigma(x) = \frac{1}{1+e^{-ax}}$ Tanh: $\sigma(x) = \tanh(\gamma x) = \frac{e^{2\gamma x}-1}{e^{2\gamma x}+1}$

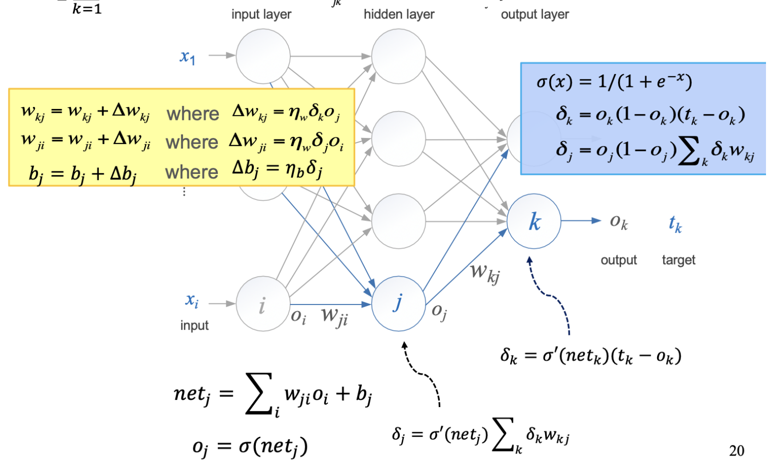
Sign: $\sigma(x) = \text{sign}(x) = \begin{cases} +1, & x \ge 0 \\ -1, & x < 0 \end{cases}$ ReLU: $\sigma(x) = \begin{cases} x, & x \ge 0 \\ 0, & x < 0 \end{cases} = \max(0,x)$

Leaky ReLU: $\sigma(x) = \begin{cases} x, & x \ge 0 \\ ax, & x < 0 \end{cases} = \max(ax, x), \quad \text{where } a \ll 1$

– Gradient Descent: $\mathbf{w}' = \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$

– Back propagation Algorithm:

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



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

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

$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial \text{net}_k}\cdot\frac{\partial \text{net}_k}{\partial w_{kj}} = \frac{\partial E}{\partial o_k}\cdot\sigma'(\text{net}_k)\cdot\frac{\partial(\sum_j w_{kj}o_j + b_k)}{\partial w_{kj}} =$
$-(t_k - o_k)\cdot\sigma'(\text{net}_k)\cdot o_j = -\delta_k\cdot o_j$

$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j}\cdot\frac{\partial o_j}{\partial \text{net}_j}\cdot\frac{\partial \text{net}_j}{\partial w_{ji}} = \frac{\partial E}{\partial o_j}\cdot\sigma'(\text{net}_j)\cdot\frac{\partial(\sum_i w_{ji}o_i + b_j)}{\partial w_{ji}} =$
$-\left(\sum_k \delta_k w_{kj}\right)\cdot\sigma'(\text{net}_j)\cdot o_i = -\delta_j\cdot o_i$

CNN:

– Stride: steps per moving. – Zero padding : pads the input with zeros around the border. – Pooling: Max: max one within filter size; Average: average within filter size.

– Regularization: $J'(\mathbf{w}) = J(\mathbf{w}) + \alpha R(\mathbf{w})$

L1 Regularization (LASSO): $R(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_k |w_k|$

L2 Regularization (Ridge): $R(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_k (w_k)^2$

Elastic Net Regularization: $R(\mathbf{w}) = \|\mathbf{w}\|_1 + \beta\|\mathbf{w}\|_2^2$

Also can be done by early stopping.

Clustering:

– Use Euclidean Distance: $d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$

– K-Means: 1. Initialize K random points as K clusters' centers. 2. Assign every point to its cluster by which center it nearest. 3. Calculate each clusters' average again to set as new center. 4. Repeat 2-3, until no points assignments. Initialization influence results.

– HAC: –Single Linkage: the minimum distance between any pair of two data samples from each cluster. –Complete Linkage: the maximum distance between any pair of two data samples from each cluster. –Average Linkage: the average distance between all pairs of two data samples from each cluster. –Centroid Distance: the distance between the means of data samples (i.e., centroids) from each cluster.

• What is the limitations of K-Means algorithm? Need to choose K. Can stuck at poor local minimum. Need good metric. • What are the limitations of HAC algorithm? Memory- and computationally-intensive.

Regression:

$f_{w,b}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

Minimize the l_2 loss: $\min_{\mathbf{w},b} \hat{L}(f_{w,b}) = \min_{\mathbf{w},b} \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{w}^T\mathbf{x}_i + b - y_i\right)^2$

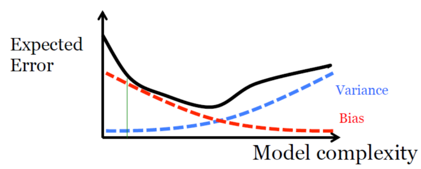
Loss function: mean squared error between $\mathbf{w}^T\mathbf{x}_i + b$ and y_i.

Bias and Variance:

– Bias: Error caused by the wrong assumptions made in the learning algorithms or models.

– Variance: Error due to the learning sensitivity to small fluctuations in the training set.

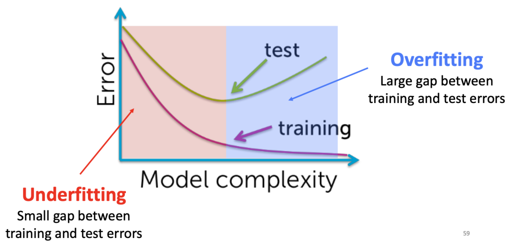
Training a classifier $f_\theta(x)$



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

– Underfitting: High bias and low variance.

– Overfitting: Low bias and high variance.



PCA:

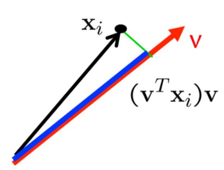
- x_i (**black**): the original data.
- v (**red**) : PCA subspace.
- $(v^T x_i)v$ (**blue**) : projected data.
- **Green:** $x_i - (v^T x_i)v$: projection error (MSE).

• Minimizing MSE <=> Maximizing Projected Variance

• Blue² + green² = **black²**

• Black is fixed (given data)

• Maximizing **blue** (variance) is equivalent to Minimizing **green** (MSE).



Bayes' Theorem:

$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$

$P(B) = \sum_i P(B \mid A_i)P(A_i)$

Naïve Bayes:

$P(a_1,\ldots,a_d \mid v_j) = P(a_1 \mid v_j)\cdots P(a_d \mid v_j) = \prod_{i=1}^{d} P(a_i \mid v_j)$