**Classification Trees**

* **Impurity:** at max. when observations are distributed evenly over all classes; at min. when all observations belong to a single class; Symmetric function of
* is the proportion of cases sent to left (l) and right (r) wrt to the total; is a function of the *relative frequencies* of the different classes in the node wrt to the split
* **Gini-index**
* :
* **Quality for a split** is the reduction of impurity it achieves:
* **Entropy:** average amount of info. generated by drawing an example at random from this node and observing its class
* **Importance measure:** of Breiman et al. may overestimate the importance of variables with low similarity to the best split but with high impurity reduction
* **Splits:** split on border of segments (,For a categorical variable with L distinct splits, there are possible splits to consider
* E.g: For a cat. var. (marital status) with 3 possible values there are: splits **to consider**: {1}, {2}, {3} 🡪 because {2,3} = {1}
* **Overfit:** model noise; to avoid: **Stopping rules** don’t expand a node if the impurity reduction is below some threshold or pruning: grow large tree and merge back
* -To prune a tree T in a node t means that t becomes a leaf node and all descendant are removed
* Definitions: means that is a pruned subtree of T; is the number of leaf nodes of the binary tree T; is the branch of tree with root note in

**Total cost of a tree**

* is the **Resubstitution error** 🡪
* is the penalty for the complexity of the tree
* Depending on , different pruned subtrees will have the lowest total cost; for the tree with smalles **Resubstitution wins,** for higher a less complex tree that makes a few more errors could win. If we continue splitting until all leaf nodes are pure T\_max = T\_1
* The total cost of T and becomes equal when 🡪 solve for
* = used to rank Trees : to rank iterate on nodes and prune nodes with minimum g value
* E.g
* When the costs are equal, the smaller tree is preferred
* Simple models tend to have high bias and low variance, Complex models the opposite
* As sample size increase, var. goes down but bias remains the same 🡪 we can fit more complex models if the data set is large 🡪 we have to find a trade-off between bias and variance in order to get small prediction error
* **Reducing variance:** By averaging, bootstrapping; Draw a sample with replacement; Grow a tree on this bootstrap sample; Repeat M times to grow M trees; Predict a test sample; Take majority voting

**Graph**

* If there is a superfluous variable in an independence, but the independence still holds without it, the independence still holds
* **Curse of dimensionality**: fewer data point than probabilities to estimate. For example the saturated model (dependent var) estimates cell probabilities by dividing cell count by total number of observations: estimation of probabilities
* **Independence model** ; Fitted count: ; estimation of probabilities
* **Independence Rules**: conditional independence or
* **Rules of probability**: summing all values of Y; marg. P of X \* cond. P of Y; if X,Y independent then 🡪
* Relax constraints: X and Y independent iff
* **Factorisation criterion for independence**:
* **Markov properties**: **pairwise**: for all non-adjacent vertices *i,j* = ; **global**, *a* separates *b* from *c*: ; **local** 🡪 boundary is the set of adjacent vertices
* product law, multiply by
* 🡪 divide by
* 🡪 in numerator we have distribution over cliques, in denominator over a subset of a clique; get fitted count by writing + jump
* = **fitted count** as ML satisfies margin constraints
* ; create table in function of **Z**, create 2 tables collapsing the independent variables; calculate fitted values
* **CPR** (degree of association) > 1 pos. association, < 1 neg. association, =1 independent!
* **Confounder term:** has to be controlled for – distorts effect of other variables 🡪
* **Bernoulli random variable** has probability p: 🡪 log linear expansion by writing the 2X2 table in log and then reparametrizing
* For 2 variable (classification problems) because if CPR=1 then logCPR=0! If then
* For 3 variables: because there cannot be terms with in common
* When we have an arbitrary number of variables then the *u* terms become functions of x rather than just constant: logP(x)
* iff all u-terms in the log linear expansion with coordinates in both b and c are zero. E.g. holds iff there are functions g and h such that: 🡪 iff u-terms with (4) and (2,5) = 0
* **A hierarchical model** is identified by listing its highest order interaction terms  
  **Graphical log-linear model**: all its constraints can be read from the independence graph; is a hierarchical model in which the highest order interaction terms correspondthe **cliques:** subset of vertices of graph such that every vertex is connected - in the graph 🡪 it gives a full characterization of the model [1, 13-16]
* **Margin constraints:**  per each clique
* **ML estimator**: returns estimates of the cell probabilities that maximize the probability of the observed data, subject to constraint that the conditional independencies expressed in the graph are satisfied by the estimates:
* **Maximum likelihood estimation of hierarchical and graphical models**: Example with 2 binary variables {0,1}   
  Usually use log-likelihood independence model has a log-linear representation as –> yield log-likelihood function applying to data available and solve it putting as constraint that cell probabilities must sum to one. Take derivative wrt Lagrange multiplier and equate to zero 🡪 solve the set of equation and find , complete the table with the fitted probabilities and obtain the fitted *u terms* from the fitted probabilities
* **Decomposable graphical Models**: have no chordless cycle – chordless only if successive pairs of vertices are connected by an edge - of length greater than three (no shortcut).
* **RIP**: E.g. {AC, BC, CDE, DEF} have {0,C,C,DE} as separators 🡪 ordering is RIP 🡪
* **Likelihood score of model** M is: (*product over all cells in the table – P of observed data using the fitted cell P according to model M*). Log-likelihood score is
* **Log linear expansion**: for 2X2 table = , independence model excludes all u-terms with interactions between
* **Model deviance** compares the log-likelihood of the fitted model with the log-likelihood of the saturated model: ; reject null hypothesis that the independence model is the true model when the observed *deviance > ,* v = df = the **difference** of the # of restrictions (**count terms**: r=rows, c=col **🡪**  how many u-terms are put to zero in case of independence) of the two models (
* where dim(M) is the number of parameters of the model
* **IPF**: sufficient statistics are row totals and column totals . 1°: Begin with table with uniform counts: puts all u terms to zero except u\_0 (which num not count)  
  2°: fit to row margins; e.g. 🡪 (*tot of row* \*   
  3°: fit to column margins; e.g. 🡪 (*tot of col*  \*
* With *k* labelled nodes there are ) undirected graphs = , edge can be excluded or included 🡪
* **Hill climbing search**: how many **neighbouring graphical models**: count edges, count new edges you can create on top of old ones, sum. **Decomposable:** count edges that can be taken away and that can be added without creating cordless cycles.

**Sets/ Pattern mining**

* **Confidence**: Conditional probability [P(Y|X)]; **Support** = relative # of X buying all items occurring in the rule [P(X)]
* **Association rule:** for association support is: ; confidence is:
* **Apriori property:** count patter only once per item
* **GSP algorithm:** level-wise search – no double count; don’ extend infrequent sequences; candidate generated iff all its subsequences are frequent [GA+GA can be GAA]
* **Lift:**  ; if *lift > 1* then rule is “interesting”.
* **Frequent set mining**: count frequency, eliminate elements which do not have support, count on all possible candidate 🡪 until no more support
* **I is Maximal frequent** iff *I* is frequentand no proper superset (set that contains it – no matter the position within the sequence) of I is frequent
* set of tuples that contain all items in I; set of items included in all transaction (which are “common elements” )in T
* . Itemset is closed iff
* If
* If
* Itemset I is a generator of a closed itemset J if there is a minimal itemset with c(I)=J

**Closures**

* **Compute closed frequent item set:** determine generators (🡪 A-priori pruning + prune if itemset has subset with same support); select all generators not pruned and determine their closure (🡪 a superset of the generator that has same support as the generator’s – if there is no closure take generator itself)**;** remove duplicate**;take** maximal frequent itemset generate subset and compute support **. A-close Apriori** is better on densely correlated data 🡪 difference between the number of frequent itemset and close frequent is higher.
* **One-to-one:** if 1-to-1, same label and order of mapping is preserved
* **Frequent tree mining: Matching functions:** denoting nodesof dby and nodes of T by  Requirements: Ancestor-descendant: ; ordering ; e.g. AI in MAIN =
* **Induced subtree =** if it is one-to-one, label preserved, left-to-right and parent-child
* **Embedded subtree=**if it has label preserved, left-to-right, ancestor-descendant (stricter than what you think)
* **Anti-monotonicity property:** D=db of trees; because ; if subsequence relation is transitive then the relation is anti-monotone wrt to support (anti-monotonicity property holds).
* **Subsequence:** if exists a mapping such thatnow assume support
* **Right-most occurrence (RMO)** list: list of nodes in the data tree to which the nodes in pattern tree can be mapped. XY graph: Y=candidate; X=# nodes

**Bayesian Networks**

* **Definitions:** then is parent of and coordinates of parents of , *i* is ancestor of *j.*Ancestors = all nodes above *(i);* descendents = all nodes below *(i);* are **marginally independent** not conditionally given the response
* **Construction of DAG**: all variables must be labelled and ordered (temporal or causal); then draw arrow for all the nodes that do not appear in the factorized joint distribution of that note. Joint distribution of is … [also factorisation of graph]. Draw arrow unless
* **Joint density:**
* **Independence properties** - **D-separation:** pathp is blocked by setZ iff: ***1.)*** P contains a chain of nodes such that B is in Z; ***2)*** p contains a collider such that neither B nor any of its descendants are NOT in Z; ***3)*** if Z blocks every path between two nodes, they are independent given Z
* **Independence properties -** you get a **Moral graph/moralization** when marry parents and delete directions of the graph edges. Factorization of moral graphs
* **Independence properties:** to determine if are independent we have to look at the “smallest marginal distribution that includes both”
* **Check conditional independencies:** to verifyalways take directed independence graph on ancestors of and moralize the graph
* **Maximum likelihood estimation:** collection of independent multinomial estimation problems: “find value of unknown parameters that maximize the probability of the observed data”. Take log and derivative wrt to p(1) of 🡪 , where
* **BN-DAG ML estimation**: with joint distribution factorization of 🡪 which is   
  Pay attention in the calculation to when variables are dependent! Directed acyclic graphs

ML estimation can be smoothed adding “prior counts” , where *m* is prior precisionis prior estimate of *p*

* **Log-likelihood score:** the higher the better the fit on data, saturated model has highest; ; recompute only changes
* **Scoring functions:** ; BIC highest penalty 🡪 less complex models preferred  
  **Count parameters**: node has *k* different parents ***configurations*** (0,1;0,0;1,0…) and take *m* values, # param. = if k=0 then # param. =   
  If asked to add a parameter, add only if variables are not independent, only in that case recompute L-score and then BIC-score
* **To score model**: Start node 1,..n,; when dependency: (1,1)-(1,2)-(2,1) …and build using
* **# of parameters of Bayesian network:** ; where *k* # var in network; #of possible values of ; # parent configuration for (1 if null)
* **Markov equivalence**: two DAGs are Markov equivalent (same score) iff have the same undirected graph when you drop the direction of all edges and same v-structures
* **Essential graph**: edge becomes bi-directional in the essential graph if there is equivalent DAG in which direction of edges is reversed
* **V-structure:** encodes the independence between two nodes
* **Bayesian network:** we pick   
  add interaction term with B (i.e.
* **Logistic regression**  ; LN {the odds, if > 1 class 1; same if class 1
* **Missing terms**: if we have three binary variables and we observe (1,0,?); in BN
* **Naïve Bayes:** compute class priors**:**  :# doc in each class; word probability 🡪
* **Link-based classification:** link attributes are based on the class labels or categories of the linked objects. When classifying new cases the problem is that the link attributes are not observed -> to predict the class of an object we need the class labes of its neighbour. Use Iterative Classification Algorithm 🡪 assing initial class label to object – apply full model to classify each object until the stopping criterion has been satisfied
* **Node-**classification problem: homophily: a link between individuals is correlated with those individual being similar; co-citation: similar individuals tend to refer or connect to the same things – the existence of an explicit link structure makes the node classification problem different from traditional data mining classification tasks (not independent!) **A clock in the middle of a watch
  
  Description automatically generatedA close up of a watch
  
  Description automatically generated**