Artificial Intelligence

**Intelligence** - characterized by:

* + Learning
  + Reasoning
  + Problem solving
  + Perception
  + Language

Learning -> Generalization: apply past exp to new situations.

Reasoning -> Ability to draw inferences appropriate to the situation

* **Deductive**: irrefutable theorems built from axioms and rules (he is either in café or bar, the bar is closed -> he is in café)
* **Inductive**: data collected that predict future behaviour, until the appearance of anomalous data forces the revision. (usually accidents are caused by men -> this accident is most likely caused by men)

Problem solving:

* In AI is a systematic search through a range of possible actions is order to reach some predefined solution.
* **Special purpose**: made for a particular problem and often exploits specific features of the situation in which the problem is embedded.
* **General purpose**: is applicable to a wide variety of problems

Perception:

* The environment is scanned and the scene is decomposed into separate objects in various spatial relationships.

Language:

* A system of signs having meaning by convention

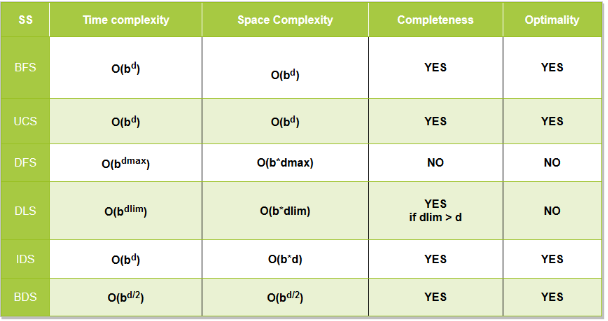
**AI Methods**

* **Symbolic** **(top-down)**: seeks to replicate intelligence by analyzing cognition independent of the biological structure of the brain. (Image processing, robotics, computer vision)
* **Connectionist(bottom-up)**: involves creating ANN in imitation of the brain structure.

**Optimisation problems:** collection of mathematical principles and methods used for solving quantitative problems in many disciplines.

**Search problems**: more actions that map the initial state of problem into a final state.

Solving the problems by search

* Computational complexity (temporal and spatial)
* Completeness (the algorithms always ends and finds a solution (If it exists)
* Optimality (the algorithm finds the optimal solution

**Search Strategies**:

* **Uninformed search strategy: (without heuristics)**
  + General, not based on problem specific information
  + Brute force methods.
  + USS in linear structures: linear search, binary search
  + USS in non-linear structures: BFS(FIFO), UCS( uniform cost search, BFS with priorities), DFS(LIFO), DLS( depth limited search, DFS with depth limit), IDS (iterative deepening depth search, IDS with depth progressively increasing), bidirectional search
* **Informed search strategy: (with heuristics)**
  + Based on specific information about the problem (trying to intelligently choose the nodes)
  + An evaluation (heuristic) function sorts the nodes
  + Specific to the problem
  + Global search strategies: Best first search (greedy or A\* + versions of A\*)
  + Local search strategies: Hill Climbing, Simulated Annealing, Tabu Search

**State space in tree-based structures:**

* H(n) = evaluation function for estimating the cost of a solution path from node A to final node B
* G(n) = evaluation function for estimating the cost of a solution path from final node B to initial node A
* F(n) = evaluation function for estimating the cost of a solution through node A = **H(n)+G(n)**

**Heuristic functions:**

* Evaluation of the state potential
* Estimation of the cost from current state to final state
* H(n) >= 0 for all state n. H(n) = 0 for final state. H(n) = infinity for a state that is dead end.
* Examples: Puzzle – h(n) = no of pieces that are in wrong places, h(n) = sum of Manhattan distance of each piece to final position). Travelling salesman problem – h(n) = nearest neighbor. Pay sum with min nr of coins h(n) = largest coin smaller than the sum to be paid

**Best-first search:**

* b – ramification factor, d – maximal depth => Time complexity = O(b^d)
* Space complexity: S(n) = T(n)
* Completeness- NO, infinite paths if each node is best selection.
* Optimality: depends on heuristic
* Advantages: Specific info helps the search, good speed to find final state.
* Disadvantages: state evaluation effort, ‘bad’ paths could seem to be good.

**Hill-climbing:**

* Initial state, final state.
* State evaluation f maximization
* Neighborhood: possible moves
* Get all neighbors then apply f and if f(neighbor) better than f(current\_state) switch.
* **Converges to local optima!!**
* Advantages: Simple implementation, does not require memory (does not comeback to previous state)
* Disadvantages: evaluation function is difficult to be approximated, the algorithm might be inefficient if there are large nr of moves. **The alg. Can block in a local optimum, or the evaluation can be constant.**
* Stochastic HC: next state is randomly selected. First-choice HC–random generation of successors until a new one is identified. Random-restart HC- restart the search from a random initial state when search does not advance

**Simulated Annealing**

* Successors of the current state are randomly selected. If a successor is better then current than swap, otherwise is retained with a given probability.
* Weak moves are allowed with a given probability p=e^deltaE/T. (escape from local optima)
* Convergence (complete, optimal) through global optima is slowly.
* Advantages: Finds optimal solution, but it requires many iterations. Easy to implement. General, if found a global solution. Can solve complex problems.
* Disadvantages: Slowly algorithm convergence to solution takes a long time. Depends on some parameters.
* Applications: combinatorial optimization problems. Design problems. Planning problems.

**Evolutionary algorithms (EA’s)**

**Design**

* Chromosome representation
* Population model
* Fitness function
* Genetic operators (selection, mutation, crossover)
* Stop condition

Chromosome’s representation:

* Linear:
  + Discrete:
    - Binary: knapsack problem (genotype-bit strings, phenotype-booleans)
    - Not-Binary:
      * Integers:
        + Random: image processing
        + Permutation: travelling salesman problem (genotype-permutation of n elements, phenotype-utility of permutation in problem)
      * Class-based: map coloring problem (genotype: vector of labels from a given set, phenotype labels meaning)
  + Continuous (real) – function optimization (genotype: vector of real numbers, phenotype: number meaning)
* Tree-based -> regression problems (genotype: trees, nodes functions, leaves terminal, phenotype: meaning of S-expressions)

**Population**

* Keep a collection of possible solutions (repetitions are allowed)
* Properties: fixed dimension u, diverse, number of different fitnesses/phenotypes/genotypes
* The entire pop evolves, not only the individuals.
* Generational EA: each generation creates u offsprings, each individual survives a generation only, set of parents is totally replaced by set of offspring.
* Steady-state EA: each generation creates a single offspring, a single parent (worst one) is replaced by the offspring.

**Fitness**

* Associates a value to each candidate solution. Reflects the adaptation to environment.
* Typology: One objective vs multi objective. Maximization vs minimization. Deterministic vs heuristic.

**Selection**

* Gives more survival chances to better individuals. Orients the pop to improve its quality
* Works at pop level. Is based on fitness only. Helps to escape from local optima.
* Typology:
  + Aim:
    - Parent selection for reproduction
    - Survival selection for next generation
  + Winner strategy:
    - Deterministic - best wins
    - Heuristic – the best has more chances to win
  + Mechanism
    - Selection for recombination
      * Proportional selection (based on fitness) – entire pop:

**+** Simple algorithm, **-** but premature convergence, **-** low selection when fitness functions are similar, **-** real results are different to theoretical prob distribution, **-** works at the entire pop level

Solution: fitness scaling

* + - * Rank-based selection – entire pop: sort entire pop based on fitness, each individual receives a rank, compute selection prob based on ranks, tries to solve problems with proportional selection by using relative fitness. **+** keep the selection pressure constant, **-** works with entire pop. Solution: another selection procedure
      * Tournament selection – on part of a pop: choose k individuals (k- tournament size), selects best individual from the sample. Probability of sample selection depends on rank of individual, sample size(k) and choosing manned, with or without replacement. **+** doesn’t work with entire pop. Easy to implement, easy to control selection pressure by using parameter k. **–** real result different from theoretical results.
    - Survival selection (selection for replacement)
      * Age-based selection (eliminate oldest individual)
      * Fitness-based selection: proportional selection, ranking selection, tournaments selection, Elitism keep best individuals from a generation to next one, Genitor: eliminate worst k individuals

**Variation operators**

* Generation of new possible solutions. Works at individual level. Must produce valid individuals. Arity 1 -> mutation. Arity > 1 -> recombination/crossover.
* Mutation: reintroduces in pop the lost genetic material. Unary operator in continuous space. Introduces diversity in population (discrete space).
  + Binary: strong mutation bit flipping, weak mutation probability mutation rate
  + Integer: random reset (a value is changed with prob p to another from the domain). Creep mutation (a value is changes with prob p to another from domain by adding a positive/negative value).
  + Permutation: swap (randomly choose 2 genes and swap their values). Insertion (choose 2 genes, insert second one after first one), inversion (choose 2 genes the range of genes are inverted), scramble, k-opt
  + Real representation: uniform (gene g is changed by prob p into a new value randomly uniform generated from a range). Non-uniform (a value is changed by adding a positive/negative value with a prob p. the added value belongs to a distribution gaussian, Cauchy, Laplace and it is reintroduced to an interval *clamping)*
* Recombination: mix the parents information. The offspring inherits from both parents and can be better weaker or the same compared to parents.
  + Binary and integer: cutting point crossover. Choose n cutting points, cut the parents put together the resulted parts by alternating the parents.
  + Binary and integer: uniform crossover: for each gene a uniform random number r is generated. If r<p will inherit from first parent, otherwise from second one.
  + Permutation: order crossover: get a substring from parent p1, copy into offspring, copy the rest of genes from p2 starting with the first pos after substring respecting genes order from p2 and no duplicates.
  + Real representation: discrete crossover: similar to uniform crossover for binary
  + Real representation: arithmetic crossover: z = ax + (1-a)y.

Singular arithmetic: choose a gene from parents same position and combine them

Simple arithmetic: select a pos k and combine all genes after that (<k unchanged)

Complete arithmetic: all genes are combined.

Geometric: each gene of child is product of parents genes each of them by a given exponent W and 1-W

**Stop condition**

* **An optimal solution was found. The physical resources were ended. The user resources were ended.**

**EAs:**

* **Complexity: most costly part fitness evaluation.**
* **General, only representation and fitness function are changed**
* **AE give better result because no linearization, not based on presumptions, doesn’t ignore possible solutions. Able to explore more possible solutions than human can.**
* **Disadvantage: large running time.**

**Particle swamp optimization (PSO)**

* Populations of particles that search the optimal solution. (approx. like EAs with chromosomes). Retains the place where it has obtained the best result. Has associated a neighborhood of particles. Exchange information among them (regarding the discoveries performed in the places already visited). Each particle knows the fitness of neighbors such as it can use the position of the best neighbor for adjusting its velocity.
* Create initial pop of particles (random positions or random velocities). Evaluate particles. For each particle update the memory (identify best particle of neighborhood and best position). Update velocity, update position. Repeat until stop.
* Each particle has: a position (possible solution of the problem), a velocity (changes a pos into another), fitness function. Each particle interacts with its neighbors, memorize a previous position.
* Principles: Proximity, quality, stability, adaptability
* **Differences from EAs: no recombination operator, position update = mutation, selection is not utilized – survival not based on fitness**
  + PSO discrete (binary) – possible solution: binary string. Velocity element from continuous space.
* Risks: rapid convergence -> local optima. Solution: re-initializate particles
* Dispersion index: spreading degree of particle around the best particle of the swarm
* Velocity index: moving velocity of the swarm into a iteration.

**ACO**

* The optimization problem must be transformed into a problem of identifying the optimal path in an oriented graph.
* Ants construct the solution by walking through the graph and put pheromones on some edges.
* Optimization based on ant colonies (=EAs)that search the optimal solution. **Cooperation instead of concurrence like in EAs.**
* Each ant moves and put pheromones in its path. Memorizes the path. Selects the path based on the existing pheromones on that path or heuristic info associated to that path. Cooperates with other ants through pheromone trail that depends on the solution quality and evaporates while time passing.
* Algorithm: initialization. For each ant increase partial solution with an element, change locally the pheromone trail based on the last element added in solution. Change the pheromone trail in paths traversed by all ants/best ant. Return the solution identified by best ant.

**Machine Learning**

* Improve a task respecting a performance metric based on experiences.
* Identify the function that must be learnt. Find a representation. Select an algorithm by using training data. Minimize error.
* Evaluate learning system:
  + Experimental: cross-validation (comparing different methods on different data); collect data based on performances (accuracy, training time, testing time); statistical analyze of the differences.
  + Theoretic: mathematical analysis of algorithm proving computational complexity, ability to match the training data , complexity of the most relevant sample for learning.
  + Comparing the performance of 2 algorithms for solving a given problem.
  + Comparing based on confidence intervals
* Choose the training database:
  + Direct: pairs (in, out) useful for the objective function.
  + Indirect: useful feedback (ex. score) for the objective function
  + Data sources: random generated, real examples, positive examples collected by learner
  + Contains training data and testing data. Independent data, respect same distribution law.
  + Quantitative characteristics: continuous(weight), discrete(nr), range (event times)
  + Qualitative characteristics: nominal(color), ordinal(intensity), structured (trees root generalization of children ex. Vehicle-> car, bus, tractor, truck)
  + Data standardization/normalization: x-std(x)/mean
* Machine learning algorithms typology (based on AIM):
  + For prediction: predict output for a new input based on previously learned model
  + For regression: estimation of the function shape based on a prev. learned model
  + For classification: classify an object into a category based on characteristics
  + For planning: generate a sequence of optimal actions for performing a task
* Machine learning algorithms typology (based on exp learned during training process):
  + With supervised learning
  + With unsupervised learning
  + With active learning
  + With reinforcement learning

**Supervised learning**

* Aim: to provide a correct output for a new entry
* Training data and testing in pairs, [input, output]. Input = (a1, a2, ...) ai attribute. Output can be a category from k classes -> classification problem, or can be a real number ->regression
* **Supervised learning = process of finding a function that maps input to output**
* There are 2 steps, learning and testing. Regression: prediction of an output for a new input (continuous) – ex prediction of prices, classification: classifying a new input (discreet) – ex detecting tumors
* The quality of learning is a measure of the algorithm’s performance.
* Evaluation methods: disjoints sets->divide the training set in 2 parts, one for learning one for testing. Cross-validation with h equal subsets -> divide data in h parts, h-1 for training 1 for testing, the performance is the average of the h runs. Leave-one-out cross-validation: similar with cross validation but h=nr of elements from data set.
* Difficulties: over-fitting->good performance on learning data, very poor on test data.
* Performance measures:
  + Statistical measures
    - Accuracy (nr. of correct / total no.), the opposite of error, determined on training set and testing set.
    - Precision: (no. of correct positive classified/ total no. of examples classified as positive) TP/(TP+FP), the prob. of a positive classified example is relevant
    - Rappel (no. of correct positive classified/total no. of positive examples) the probability that a positive example is correctly classified. TP/(TP+FN). Confusion matrix – real results vs computed results.
    - Score: combines precision and rappel. 2PR(P+R)
  + Efficiency
  + Scaling
  + Compactness
  + robustness

**Unsupervised learning**

* Aim: finding a model or structure in data set. Dividing a set of unlabeled examples in disjoint sets. The elements of a cluster are very similar. The elements of diff. cluster are different.
* **Unsupervised learning: process of finding a function that groups the elements from the data set in several classes. The nr. Of classes can be predefined or unknown. The elements from one class are similar according to some criteria.**
* Distance – min, similarity – max
* Training – learning the clusters. Testing – testing the model by placing a new element in a cluster.
* Quality of learning:
  + Internal criteria: high similarity in clusters, reduced similarity from different clusters
    - Distance inside the clusters: average distance between points, average of distances between the closest 2 points, distance between centroids.
    - Distance between 2 clusters: simple link (min(||xi1\*xj2||) for xi1 in c1 and xj2 in c2), complete link(max(||xi1\*xj2||). Average link, centroid link.
    - David-Bouldin index: min -> compact clusters
    - Dunn index: identifies compact and well separated clusters D = Dmin/Dmax
  + Classification criteria: the clustering determination
    - Hierarchic: taxonomic tree, agglomerating (from small clusters to big ones), divisive (from large clusters to small ones)
    - Non-hierarchic: partitioning->dividing data set->all clusters simultaneous, optimizes an objective function, *no of clusters is predefined*,
    - Based on data density: the clusters creation is based on data density in some region, the clusters creation is based on data connectivity is some region => the clusters can have any shape
    - Grid based: the space is segmented in regular zones. The objects are placed in a multidimensional grid (ex. ACO)
  + Classification criteria; typology
    - Considering the algorithm behaviour:
      * Agglomerating: every element is a cluster. Compute distance between each 2 clusters. The closest 2 merge. Repeat until stop.
      * Divisive: random k clusters. Divide elements to k clusters. Recompute new centers for the clusters. Repeat until the algorithm converges (no change)
    - Considering the attributes
    - Considering the type of membership of elements to clusters.
      * Hard clustering
      * Fuzzy clustering

**Active learning: the algorithm receives supplementary information during learning faze in order to improve its performances.**

**Reinforcement learning: the main feature is the interaction with the environment (actions->rewards)**

**Automatic learning:**

* Least square
* Descending gradient
* Evolutionary algorithms
* kNN
* Decision trees
* Support vector machines
* ANN
* Genetic programming
* Hidden Markov models.

**Least Square method:**

* Consider a regression problem.
* Input data (x) and output data (y)
* Determine a model f that maps x in y
* F(x) = b0 + b1\*x1 + b2\*x2 + … + bn\*xn
* Loss function: Sum(yi – f(xi)^2) : minimize, obtain optimal values for b

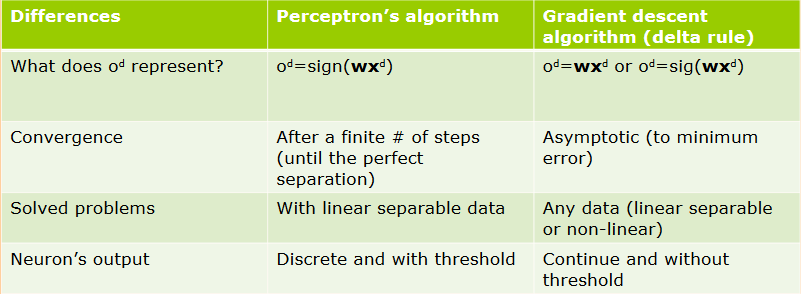
**Descending gradient method**

* Iteration 0: random b values.
* Iteration t+1: b(t+1) = b(t) – learningRate\*error(t)\*x
* Error = computed – realOutput.

**Decision trees:**

* Divide a collection in smaller sets by successively applying some decision rule (asking q)
* A graph with 3 nodes. Decision nodes, hazard nodes and result nodes (leaves). Each node is an attribute, each branch under a node is a value of that attribute. Each leaf is a class.
* Fixed nr of attributes, each attribute finite nr of values. Objective function takes discrete values. Training data could contain errors or could be incomplete. Can be used in classification problems or regression problems.
* Process:
  + tree construction: based on training data – bottom up or top down: Split the training data into subsets based on characteristics.
    - Attribute selection: select the attribute that corresponds to a node:
      * random,
      * fewest/most values,
      * information gain,
      * gain rate,
      * Gini index
      * distance between partitions created by the attribute.
  + use tree as problem solver – rules from dt are used for labeling new data.
    - Difficulties: underfitting (training data too simple) -> large classification error on training and testing. Overfitting: good on training bad on new data
    - Solutions: pruning (remove some branches) -> small tree. Cross-validation
  + Pruning: identify and eliminate branches that reflect noise or exceptions
    - Pre-pruning: increasing the size of three is stopped during construction by stopping division when majority class of examples from that node
    - Post-pruning: After dt is constructed, eliminate branches of some nodes that become leaf
* Advantages: easy to understand and interpret. Can use nominal or categorized data. Works better with large data. Decision logic can be easy followed (rules are visible)
* Disadvantages: instability – change the training data. Complexity -> representation, difficult to use, the dt construction is expensive and requires lot of information.

**ANN – Artificial neural networks**

* Aim: binary classification for any input data
* A structure similar to biological NN. A set of nodes (neurons) located in a graph with more layers. Nodes have inputs and outputs. Connected by weighted links that influence the performed computations. Nodes perform a simple computing through activation functions. Layers: input layers – contains m nodes (nr. attributes of a data). Output layer: contains r nodes (nr. of outputs). Intermediate layers: different structures, different sizes.
* A training set. Form an ANN with input nodes and output nodes and add hidden layers. Determine the optimal weights by minimizing the error.
* Neurons performs simple computation: information is transmitted to the neuron -> compute the weighted sum of inputs then processes the information by using an activation function:
  + Constant function
  + Step function
  + Slope function
  + Linear function
  + Sigmoid function
  + Gaussian function
* Read the neurons answer -> determine if the computed result is the same with the real one
* Neuron learning:
  + Perceptron’s rule: start with random weights, determine the quality of these weights for a single input data, recompute weights, repeat until maximum quality is obtained.
  + Delta’s rule: start with random weights, determine quality for all input data, recompute weights, repeat until maximum quality is obtained.
  + Gradient descent algorithm: modify weights in the direction of the steepest slope or error reduction. The steepest slope is determined by deriving E based on w. Errors gradient computation. Linear or sigmoid function. Simple GDA vs Stochastic GDA ( stochastic changes the weights by adding the error to the old weight)
  + Backpropagation: distribute the errors on all connections proportional to the weights and modify the weights. Backpropagation stops when error is 0.
* Typology
  + Feed-forward ANN: info is processed and passed from a layer to another layer. Node connections do not form cycles. Utilized for supervised learning, activation functions linear, sigmoid and gaussian.
  + Recurrent ANNs (with feedback): can contain connections between nodes of the same layer. Node connections can form cycles. Jordan, Elman, Hopfield ANNs for supervised learning, self-organized ANNs for unsupervised learning. (Hebbian, Kohonen)
* Advantages: can solve supervised and unsupervised learning problems. Can identify dynamic and non linear relations among data. Can solve multiclass classification problems. Can compute very fast.
* Disadvantages: over-fitting problems, even if cross-validation is performed. Can stuck at local optima.

**Deep learning**

* Sub field of machine learning similar to ANN. Similar to the functioning of our brains.
* Deep convolutional Neural Networks: used to classify images, cluster them by similarity, perform object recognition within scenes Algorithms that can identify faces, individuals, street signs, tumors, perform optical character recognition.
* Tensors: (rank n) a mathematical object in m-dimensional space that has n indices and m^n components and obeys certain transformation rules. It is a generalization of scalars, vector and matrices.
* Convolution operation: s(t) = (x\*w)(t) or SUM(x(a)w(t-a)) a from -inf to inf
* In practice we have 2 tensors, input ( a multidimensional array of data), the kernel(a multidimensional array of parameters). Stored separately->these functions are zero everywhere but in the finite set of points.
* Images as tensors. Reduce the images without losing critical features.
* Using a kernel, we want to apply a filter over the image, and extract the high-level features. (edges, color, gradient, orientation).
* Layers: 3 stages: first: several convolutions in parallel. Second a nonlinear activation function ->detector stage, third: pooling function to modify the output.
* Pooling: helps to make the representation approximately invariant to small translations of the input. Useful property if we care more about whether a feature is present than where it is. Essential for handling inputs of varying size. Max pooling: returns the maximum value from the portion of the image covered by the Kernel. Average pooling returns the average of all the values from the portion of the image covered by the kernel.

**KBS**

* Information about domain in a given representation. Required information for problem solving. Set of sentences that describes an environment.
* Typology: perfect knowledge, imperfect knowledge
* Knowledge representation:
  + formal logic: science of formal principles for rationing. Components: syntax (atomic symbols used for constructing rules), semantic (associates a meaning to each symbol and a truth value to each rule), syntactic inference (rules for identifying a subset of logic expressions->theorems)

Typology: true value: dual, polyvalent. Basic elements: sentences or random variables. Working manner: propositional logic, first order logic.

* + Rules: special heuristics that generate information. Inter-dependencies among rules. Link the cause and effect -> if clause then effect
  + semantic nets: oriented graph with nodes(concepts), edges (semantic relations among concepts). Meronymy (A is meronym of B if A is part of B). Holonymy (A is holonym of B if B is part of A), Hyponymy (A is hyponym of B is A is a kind of B), hypernymy(A is a hypernym of B is A is a generalization of B), synonymy(A is a synonym of B is A and B have the same meaning), antonymy(A is an antonym of B is A and B are opposite).
* Inference engine: determine a conclusion by taking into account some premises and by applying some inference rules.
* Inference direction: forward chaining: start from available info and draw a conclusion. (data driven) Backward chaining – start from a conclusion and identify some explanations for it (goal driven)
* Techniques for inference. Logic based vs rule based in certainty environment, based on theory of probability vs possibility in uncertainty environment (fuzzy).

**Logic based systems (LBS)**

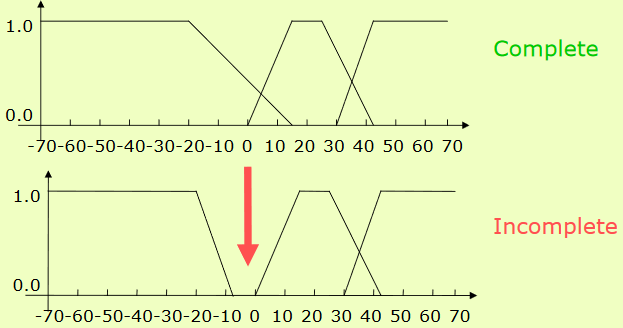
* Explore a lot of info for obtaining conclusions about difficult activities by using methods of formal logic. Formal logic is precise and well-defined.
* Systems based on propositional logic: Declarative propositions only, described object are unique and fixed.
  + Inference engine: Is it possible to deduce O from KB? Model checking: construct the table for all possible combinations of truth values for all the symbols. Determine if all the models of KB are model for O too. KB models, models where all propositions are true.
  + Difficulties: exponential number for all combinations -> large computing time
  + Solution: deduction by using inference rules. Construct a proof of truth value for the objective proposition based on propositions and inference rules.

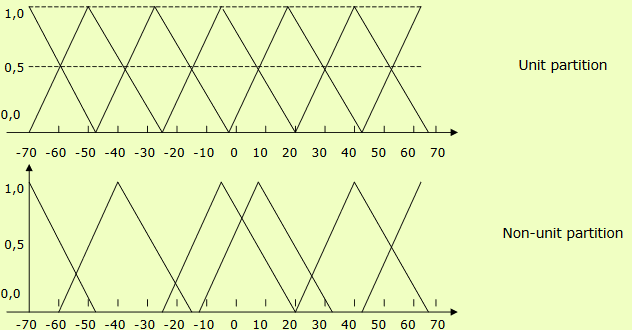
**Rule based systems (RBS)**

* Explore a lot of knowledge for new conclusions about activities that are difficult to be analyzed by using methods as humans do. Can solve problems for which there is no deterministic algorithm. Solving problems that require human experts. (ex. Microsoft windows troubleshooting, financial consulting, medical diagnosis, car diagnosis)
* Knowledge necessary for understanding, formulating and solving problems. (facts, rules)
* Storing all knowledge about a specific domain taken from human experts or other sources.
* Facts: correct affirmations, memorized as some data structures.
  + Typology:
    - persistence (changing rate) static facts, dynamic facts (Bob takes a lunch break – specific for an instance)
    - generation: given facts, derived facts.
* Rules: special heuristics that generate knowledge
  + Typology:
    - Certainty degree: exact rules, uncertain rules(if winter, temp < 0)
    - Rules can express: relations(if he is good he has friends), recommendations(if it rains, should get an umbrella), directives(if no more battery, must charge), heuristics(if phone sound is off maybe phone dead)
  + Rules advantages: easy to understand, easy to derive inference and explanations, easy to modify and maintain.
  + Rules limits: complex knowledge requires many rules. Search limitations in systems with many rules.
* Inference engine: rules that can generate new info, algorithms, rbs’s brain a deduction algorithm based on KB and specific to reasoning method.
  + Typology:
    - Forward chaining: start with initial facts and keep using rules to draw new conclusions. Rules of the form left-hand-side => right-hand-side. Data-driven start from available info as it becomes available then try to draw conclusion
    - Backward chaining: get a state check if initial facts matches, if not check rules.
* Forward chaining should be used when all or most data is given in problem statement. Great nr of goals but few achievable. It is difficult to formulate a goal or hypothesis.
* Backward chaining should be used when a goal or hypothesis can be easily formulated. Large nr of rules that match facts => large nr of conclusions. Problem data are not given.
* Conflict resolution! – More than one rule is matched
  + If color is yellow => apple.
  + If color yellow and shape longish => banana
  + If shape is round => apple
* Approaches: first rule, random rule, most specific rule, lest recently used rule, best rule (prioritization)

**KBS – Fuzzy Systems**

* Generalization of Boolean logic. Deals with the concept of partial truth. Classic logic true = 1, false = 0, in fuzzy logic gradual expression of truth with values between 0 and 1.
* Conjunction = minimum => a ^ b = min(a, b). Disjunction = maximum a v b = max(a, b). Negation = difference => not a = 1 – a
* Constructing a fuzzy system: Define inputs and output, get the raw inputs and outputs a fuzzify them. Create a base of rules. Evaluate the rules – inference. Aggregate the results, defuzzificate the results and interpret them.
* Fuzzy facts
  + Set definition
    - By enumeration of elements
    - By specifying a property of elements
  + Characteristic function u for a set: 1 = belongs, 0 - not belongs, otherwise a part of F
* Fuzzy sets:
  + Singular: u(x) = s, s scalar.
  + Triangular: MAX(0, min(x-a/b-a, 1, c-x/c-b)
  + Trapezoidal: MAX(0, min(x-a/b-a, 1, d-x/d-c)
  + Z function: u(x) = 1-S(x)
  + PI function => u(x) = S(x) if x<=c otherwise Z(x)
* Operations:
  + Complement: B is complement of A if ub(x) = 1 – ua(x) for all x from X
  + Containment: B is a subset of A if ub(x) <= ua(x) for all x from X
  + Intersection: C is a intersection of A and B if uc(x) = min(ua(x),ub(x)) for all x from X
  + Union: C is a union of A and B if uc(x) = max(ua(x),ub(x)) for all x from X
  + Equality: B is equal to A if ub(x) = ua(x) for every x from X
  + Algebraic product: C is the product of A and B if uc(x) = ua(x)\*ub(x) for all x from X
  + Algebraic sum: C is the sum of A and B if uc(x) = ua(x)+ub(x) for all x from X
* Fuzification properties:
  + Associativity
  + Commutativity
  + Distributivity
  + Transitivity
  + Idem potency
  + Identity
  + Involution
* Hedges – modifiers, adjectives or adverbs that change the truth values of sentences. Heuristics. Can act on fuzzy numbers, truth values, and membership functions.Subjectivism.
* Very u(x)^2, Extremely u(x)^3, Very very = u(x)^4, Somewhat u(x)^1/2, slightly u(x)^1/3, indeed intensifies truth value if truth <0.5 => 2u(x)^2, otherwise for truth >0.5 1-2(1-u(x))^2
* Fuzzy variables: defined by V = {x,l,u,m}
  + X – name of symbolic variable (ex. temperature)
  + L – set of possible labels for variable x (ex. Low, medium, high)
  + U – universe of the variable (ex. -70 <= x <= 70)
  + M – semantic regions that define the meaning of labels from L (graphical repres.)
* Fuzzy variables – properties:
  + Complete/Incomplete – V is complete if for all x there is a fuzzy set such that ua(x)>0
  + Unit/Non-unit partition: V forms a unit partition if for all x we have SUM(ua(x)) = 1

a complete fuzzy variable can be transformed into a unit partition



* Fuzzification of data: transform each raw input data into a linguistic data
* Rules: non-conditional vs conditional. If A then B. A premise, B consequence.
* Rules properties: Completeness, a db of fuzzy rules is complete. Consistency, A set of fuzzy rules is inconsistent if 2 rules have the same premises and diff consequences. Problems => nr of rules increases exponential with the nr of input variables. Nr of input set combinations is products of variable sets.
* We also have a decision matrix that is constructed based on rules.
* Fuzzy inference: forward inference, backward inference.
* A rule can have more premises linked by logic AND, OR, NOT.
* Evaluation of consequences: determine the result, defuzzify it to obtain crisp value.
  + Mamdani model: output variable belongs to a fuzzy set.
    - Clipped fuzzy set: membership function of the consequence is cut at the level of truth value result. + easy to compute, - info is lost
    - Scaled fuzzy sets: membership function is adjusted by scaling at the level of truth value, + few info lost, - complicate computing
  + Sugeno model: output variable is a crisp function that depends on inputs.
  + Tsukamoo model: output variable belongs to a fuzzy set following a monotone membership function.
* Aggregate the result: union of outputs for all applied rules. Different at Mamdani model vs Sugeno model.
* Defuzzificate result: transform the fuzzy result into a raw/crisp value. Methods:
  + Based on the gravity center:
    - COA (centroid area)
    - BOA(bisector of area)
  + Based on maximum of membership function
    - MOM – mean of maximum
    - SOM – smallest of maximum
    - LOW – largest of maximum
* Interpret the result
* Advantages: easy to understand, test, maintain. Robustness, can operate when rules are not so clear. Require few rules then other KBS, rules are evaluated in parallel. Imprecision and real world approx.. can be expressed through some rules.
* Disadvantages: require many simulations and test. Do not automatically learn. It is difficult to identify the most correct rules. There is no mathematical model.

**Genetic programming (GP)**

* A special case of evolutionary algorithms
* Chromosomes: tree with nodes of type function and terminal. Leaves = terminals. The tree encodes the mathematical expression of the program.
* Fitness: prediction error - difference between what we want to obtain and what we actually obtain.
* Chromosome initialization – random generation of correct trees -> valid programs. Establish a maximal depth of the trees. 3 initialisation methods:
  + Full => each branch of the root has depth Dmax. Nodes of depth < Dmax are initialized by a function from F. Nodes of depth = Dmax are initialized by a terminal from T.
  + Grow => each branch of the root ha s depth <Dmax. Nodes of depth <Dmax are initialized by an element from F u T. Nodes of depth =Dmax are initialized by a terminal from T.
  + Ramped half and half. ½ pop is initialized by full and ½ of pop by grow.
* Recombination selection: over-selection: for very large populations. Sort the pop based on fitness and consider 2 groups, group 1 contains the best x% chromosome from population. Group 2 contains (100-x)% chromosomes form population. 80% of selection operators will choose chromosome from the first group and 20% from the second group.
* Survival selection: generational, steady-state. Problems, bloat = the fattest individual survives, solution, block the variation operators that produce the fat offsprings. Parsimony pressure = to give a penalty in the fitness function to long programs or program with many non-coding parts.
* Probability of choosing XO and mutation Koza (p=0), Banzhaf (p=0.05).
* Size of offsprings can be different to size of parents.
* Crossover: By cutting point. Subtrees are exchanged. Cutting point is randomly generated.
* Mutation:
  + grow mutation – replace a leaf by a new subtree.
  + Shrink mutation – replace a subtree with a leaf.
  + Koza mutation – replace a node by a subtree
  + Switch mutation – select an internal node and reorder its subtrees.
  + Cycle mutation – select a node and replace it by a new node of the same type (internal node with a function, leaf node with a terminal)
* GAs vs GP
  + GAs linear chromosomes, GP non-linear chromosomes.
  + GAs fix chromosome size, GP variable chromosome size.
  + GAs offspring generation XO and mutation, GP XO or mutation
* Advantages: GP finds solutions for problems without an optimal solution. Gp is useful for problems whose variables are frequently changed.
* Limits: Large time required for evolving the solution.
* GP versions:
  + Linear GP – fitness computation does not require interpretation -> works fast.
    - Representation: vector of statements
    - Initialization: random or constraints (initial length of chromosome)
    - Crossover – 2 – cutting points
    - Mutation – micro-mutation = change an operand or operator without modifying the size of chromosome. Macro-mutation = insert or eliminate a statement. (modifying the size of the chromosome)
    - + evolution into a low-level language. – number of required registers
  + Gene expression programming (GEP): linear representation of expression that can be encoded in a tree by bfs)
    - A chromosome is composed by more genes linked by + or \*. Each gene is composed by head(contains functions and terminals) tail(contains t terminals only where t = (n-1)\*h+1 where n maximal rarity of a function F)
    - Initialization randomly by elements from F and T.
    - Crossover at allele level one cutting point, two cutting points. At gene level chromosome exchange some genes
    - Mutation: at allele level change an element from head or tail
    - Transpositions: the introduction of an insertion sequence somewhere in a chromosome.
    - Advantages: coding into chromosomes some correct programs due to gene splitting in head and tail.
    - Disadvantages: Multi gene chromosomes, how many? How to link?
  + Multi Expression Programming (MEP)
    - Chromosome is composed by more genes. Each gene being a 3 address code. Similarly to GEP but faster.
    - Representation: Linear, a gene contains a binary or unary function and pointers to its arguments. Chromosome encodes more possible solutions. Each solution corresponds to a gene. Quality of a solution = sum of differences between what we want to obtain and what we obtain.
    - Initialisation: First gene must be a terminal. Other genes can contain a terminal or a function and pointers to its arguments.
    - Crossover: exchange some genes between parents. 1-cutting point, 2-cutting points, uniform.
    - Mutation: modify a gene. First gene randomly generate a new terminal. Other genes randomly generate a terminal or a function.
    - Advantages: Dynamic output for each chromosome.
    - Disadvantages: complexity of decoding for unknown training data
  + Grammatical evolution (GE)
    - Evolving programs in Backus-Naur form (program expressed as a grammar with terminal symbols, non terminals, start symbol and rules)
    - Representation: binary string of codons groups of 8 bits -> rule that must be applied.
    - Initialisation: binary string is randomly initialized by 0 or 1 -> valid programs. Decoding ends when a complete program is obtained. If the codons end and the program is incomplete, restart the codons from beginning -> wrapping.
    - Crossover: cutting point
    - Mutation: probabilistic change of a bit into its opponent.
    - Duplication: A sequence of genes is copied to the end of chromosome.
    - Pruning – elimination of unused genes.
    - Advantages: Evolving programs written in languages whose statements can be expressed as BNF rules. Representation can be changed by changing the grammar.
    - Disadvantages: infinite wrapping -> limit the repetitions and penalize the chromosomes that overpass a given threshold of repetitions.
  + Cartesian Genetic programming
    - Chromosomes as graphs (matrix) => more complex programs.
    - Representation: cartesian system, matrix of nodes. A node has associated a function, input and outputs.
    - Initialisation: randomly. Inputs of any node must be nodes from previous columns. Nodes of the first column has as inputs the problem attributes.
    - Crossover: is not applied.
    - Mutation: modify the elements of a node.
    - Advantages: Evolving the index of node that provides the output of the program encoded into the chromosome. Evolved program can have one or more outputs.
    - Disadvantages: number of columns influences the results.

**Support vector machines (SVM)**

* SVM finds a linear function f(x) = <w\*x> + b. w-weight vector
* <w\*x>+b = 0 -> decision hyper-plane that separates the two classes.
* There are more hyperplanes, SVM searches the hyperplane with the largest margin (that minimizes the generalization error)
  + SMO(sequential minimal optimization) algorithm
* Solved problems?
  + Classification problems
    - Linear Separable => error = 0
    - Linear Non-separable -> constraints are relaxed, some errors are allowed. Penalization coefficient – C.
    - Non-linear separable: input space is transformed into a space of more dimensions by using kernel function, in this new space the data becomes separable. In SVMs the kernel function computes the distance among 2 points.
      * Classic kernels: Polynomial kernel , RBF kernel
      * Multiple kernels: linear, non linear with or without coefficients.
      * Kernels for strings
      * Kernels for images
      * Kernels for graphs
* SVM parameters:
  + Penalization coefficient C – small->slowly convergence, large->fast convergence
  + Kernel parameters
    - If nr of attributes larger than nr of data-> SVM by a linear kernel
    - If nr of attributes is large and nr of data medium-> SVM with gaussian kernel
    - If nr of attributes is small and n large, add new attributes and than SVM with linear kernel.
* Simple SVM f:X->R
  + Any type of inputs, numerical outputs
* Structured SVM f:X->Y
  + Any type of inputs, any type of outputs
* Advantages: Can work with any type of data, linear, nonlinear, uniform distributed or not. If the problem in convex the SVM finds a unique solution -> global optima. Avoids over fitting. Automatic selection of the learn model.
* Disadvantages: Real attributes only, binary classification problems only. Difficult mathematical background.

In plus:

Cautarea locala simpla

Hill climbing – allege cel mai bun vecin

Simulated Annealing – allege probabilistic cel mai bun vecin

Cautare tabu – retina lista solutiilor recent vizitate

Cautarea locala in fascicol (beam local search) – se retin mai multe stari ( o populatie de stari)

EA, PSO, ACO

EA

Modele de populatii:

* generational adica fiecare individ supravietuiesto o singura generatie. Multimea parintilor e inlocuita in intregime cu multimea descendentilor.
* Steady-state: din fiecare generatie se obtine doar un descendent iar parintele cel mai slab e inlocuit.
* Discrepanta intre generatii 1 pentru modelul generational. 1/nrpop pentru steady-state.

Exemple

Problema rucsacului reprezentare->liniară discretă binară, fitness->abs(greutatea rucsacului – greutatea obiectelor alese) ->minimizare

Problema plăţii unei sume folosind diferite monezi reprezentare->liniară discretă întreagăπfitness->abs(suma de plată – suma monezilor selectate)->minimizareν

Problema comisului voiaiorπreprezentare->liniară discretă întreagă sub formă de permutareπfitness->costul drumului parcurs->minimizareν

Problema optimizării funcţiilorπReprezentare->liniară continuă realăπfitness->valoarea funcţiei ◊minimizare/maximizareν

Calculul ariei unui cercπreprezentare->arborescentăπfitness->suma pătratelor erorilor (diferenţelor între valoarea reală şi cea calculată pe un set de exemple)->minimizare

Selectia: de doua tipuri:

Selectia parintilor (din generatia curenta) pentru reproducere

Selectia supravieturitorilor (din parinti si descendenti) pentru generatia urmatoare.

Mutatia: la reprezentarea intreaga e random resetting si creep mutation.

Mutatia neuniforma reprezentarea reala; valoarea unei gene este schimbata cu prob p prin adaugarea unei valori (pozitiva sau negative). Valoarea face parte dintr-o distributie. Gaussiana, Cauchy sau Laplace. Si readusa intr-un interval daca este necesar (clamping)

