Computer science curricula 2013

Tom Rochette <tom.rochette@coreteks.org>

August 30, 2025 — 861fb9d0

In the following document, I try to explain to the best of my knowledge various topics that are presented in the Computer Science Curricula 2013. I attempt to rate my understanding of the topic on a scale of 1 to 5, where 1 is "no understanding" and 5 is "perfect understanding". This value is displayed in bracket in front of each statement I make.

1 Algorithms and Complexity

1.1 Fundamental Data Structures and Algorithms

1.1.1 Graphs and graph algorithms

1.1.1.1 Shortest-path algorithms (Dijkstra's and Floyd's algorithms)

[3] In a weighted (directed or undirected) graph, Dijkstra's algorithm initializes all nodes with an infinite value and the starting node with a value of 0 for the initial node. Mark all nodes (except the starting node) as unvisited. For all the adjacent nodes, compute their distance to the current node and replace their distance if it is smaller than the existing marked distance. Once all the adjacent nodes for the current node have been visited, proceed to mark the node as visited. If we have reached the target node, we're done. Otherwise, pick the node with the smallest distance as current node. and continue the process of visiting adjacent nodes.

[?] Floyd's algorithm

2 Intelligent Systems

2.1 Fundamental Issues

2.1.1 What is intelligent behavior?

2.1.1.1 The Turing test

[4] The Turing test is a simple test which purpose is to attempt to establish whether an agent can pass as a human being when tested by another agent (generally a human being).

A tester is put in front of a computer in which he can communicate with the agent/person at the other end. This agent may be a human being, or a computer program that attempts to pass as a human being. If the tester is unable to tell whether the agent at the other end of the communication channel is human or machine, then this agent has passed the Turing test (and based on this test, can be considered to have human-like communication properties).

2.1.1.2 Rational versus non-rational reasoning

[2] Rational reasoning is when something can be explained by a sequence of finite explanations based on logic, one leading to the other, similar to a proof.

[2] Non-rational reasoning is when something is explained through means such as probabilities.

2.1.2 Problem characteristics

2.1.2.1 Fully versus partially observable

[4] Fully observable means that the agent can see the whole state of the environment.

[4] Partially observable means that the agent can only see a part of the state, the state may be noisy or inaccurate due to the agent's sensors.

2.1.2.2 Single versus multi-agent

[4] In the single agent case, the agent is evaluating itself only against the environment.

[4] In the multi-agent case, the agent has to consider the actions of other agents on itself and modify its behavior accordingly.

2.1.2.3 Deterministic versus stochastic

[4] An environment may be deterministic, which means that given a known state and action, the agent can determine the final state after the action has been executed.

[4] On the other hand, a stochastic environment is one in which given a known state, the action that is taken may be randomly selected, or the result of a selected action may result in a random final state.

2.1.2.4 Static versus dynamic

[4] When the agent is thinking, it may consider that the environment isn't changing. In other words, thinking is done atomically.

[4] When the agent is thinking, the environment is changing. This means that the agent may make a decision that isn't the best given the time it took to compute it.

2.1.2.5 Discrete versus continuous

[4] A problem may be composed of discrete steps, which means that only certain values on a continuous axis are possible. In other words, there is a finite precision to the values a state can take.

[3] A problem may be described using continuous values, meaning that there is an infinitesimal distance between two points.

2.1.3 Nature of agents

2.1.3.1 Autonomous versus semi-autonomous

[4] Autonomous: An agent that makes decisions without the need of intervention from the part of another agent.

[3] Semi-autonomous: An agent that can make decisions by itself, but may also require the intervention of another agent in certain instances.

2.1.3.2 The importance of perception and environmental interactions

[3] Without perception, an agent cannot interact with its environment. Given an agent can be represented as a function, it would be similar to this function not receiving any argument (observation about the environment state). Agents that do not interact with the world are generally of little value (other than theoretical/experimental).

2.2 Basic Search Strategies

2.2.1 Problem spaces (states, goals and operators), problem solving by search

[4] A problem space is a graph that represent all the possible states that a problem can be into. An example of a state is a chess board configuration.

[3] A state is the representation of all the critical elements of the problem. In a problem, it is important to ignore the facts that are of no importance, such as the color of the cells of the chess board.

[2] In the problem graph, one can transition from one state to another through operators.

[3] Problem solving by search is the idea of constructing a graph and using the tools provided by graph search algorithms in order to solve a problem.

2.2.2 Factored representation (factoring state into variables)

[1] I couldn't find anything relevant about factoring state into variables on the Internet.

2.2.3 Heuristics and informed search (hill-climbing, generic best-first, A*)

[2] Heuristics are "decisions" or strategies that help guide the behavior of a given algorithm.

[2] When using informed search algorithms, the hope is that we can reduce the amount of tests necessary to find a good solution by making use of the additional information.

2.2.4 Two-player games (introduction to minimax search)

[2] In two-player games, the purpose of each player is to win. The best strategy, given that you expect your opponent to play optimally (as you would as well) is to pick the action that will maximize some gain. At the same time, you want to minimize the expected gain of your opponent.

2.2.5 Constraint satisfaction (backtracking and local search methods)

[3] In problems which have constraint, one way to generate a solution is to construct the state tree, and whenever a constraint is violated, to "backtrack" (more back one step) and try an alternative action.

[2] Local search methods will generate an initial assignment, and given that assignment, will try to "correct" the constraint violations by applying legal operations to the state. As operations are performed, the "state pointer" moves from one state to an associated one. Unlike in backtracking, the chain of constraints that have been explored is not fixed.

2.3 Basic Machine Learning

2.3.1 Definition and examples of broad variety of machine learning tasks, including classification

[2] Regression is the task of determining the degree of association between variables.

[3] Classification is the task of determining to which class a set of observations belongs. Regression is generally used to separate classes in a one-vs-rest fashion.

[2] Examples of machine learning tasks are: generating speech given text, generating text given speech, creating a sentence to describe an image, generate music given a few parameters, extract text from an image, etc.

2.3.2 Inductive learning

[4] Inductive learning is the process of learning from a set of examples. Given that we have no idea what the function that generated some outputs given inputs, the task of inductive learning is to construct such function using the examples it is given.

2.3.3 Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees

[1] A Naive Bayesian classifier

[3] Decision trees make use of the features (and their frequency) to determine the most optimal way in which to ask questions about these features in order to better separate the different values it can output. In order to construct the tree, some metric is used to determine the "gain" that would be generated if a given decision were to be the node under consideration.

2.3.4 Measuring classifier accuracy

[3] The measure of a classifier accuracy is the sum of all the correct classification divided by the number of examples given (incorrect + correct).

2.4 Advanced Search

2.4.1 Constructing search trees, dynamic search space, combinatorial explosion of search space

[1] Constructing search trees

[1] Dynamic search space

[3] One major issue in search is that given n options, each level of the search tree grows is multiplied by n, thus making the number of states grow exponentially. This in turn means that problem of relatively small size can be tackled without too much problem, until a certain point at which they become impossible to deal with due to the shear amount of states that would have to be scanned.

2.4.2 Stochastic search

2.4.2.1 Simulated annealing

[2] Simulated annealing uses the concept of temperature to stochastically try various state and evaluate them, probabilistically moving to an alternative state. As the temperature decreases, the probability of trying an alternative state than the currently best found decreases.

2.4.2.2 Genetic algorithms

[2] Genetic algorithms combines pairs of states to generate a new state by using both the operation of mutation and crossover. A mutation is a random change within the state while a crossover is the transfer of part of the state from a parent.

2.4.2.3 Monte-Carlo tree search

[2] Monte-Carlo tree search is the exploration of the search tree by randomly exploring the search tree most promising actions. Instead of constructing the whole tree (which may not be possible), the algorithm explores the most promising branches by simulating a possible traversal (from root to leaf). Given the result of this simulation, it is recorded as part of the branch being simulated, such that we have an idea of the potential (in term of probabilities) of each branches.

2.5 Reasoning Under Uncertainty

2.5.1 Random variables and probability distributions

2.5.1.1 Axioms of probability

[1] 3 axioms (with sample space Ω, event space F and probably measure P), also known as Kolmogorov axioms

- The probability of an event is a non-negative real number:

$$P(E) \in \mathbb{R}, P(E) \geq 0 \qquad \forall E \in F$$

  where F is the event space
- The probability that at least one of the elementary events in the entire space will occur is 1

$$P(\Omega) = 1$$

- Any countable sequence of disjoint sets (mutually exclusive events) $E_1, E_2, \ldots$ satifies

$$P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i)$$

2.5.1.2 Probabilistic inference

[1] Given a set of variables (and their associated probabilities), determine the probability distribution of a query variable, that is, a variable which can be expressed as the combination of the given set of variables.

2.5.1.3 Bayes' Rule

[2]

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

where A and B are events and $P(B) \neq 0$.

2.6 Conditional Independence

[1]
Given three events, A, B and C, with B and C being conditioned on A, B and C being conditionally independent means that given A occurs, if B occurs, we cannot tell anything about C and vice-versa.

2.7 Agents

2.7.1 Definitions of agents

[3] An agent is an entity that interacts with an environment and possibly other agents. In artificial intelligence, an agent can be categorized under 4 different types:

- Reflex based
- Model based
- Goal based
- Utility based

2.7.2 Learning agents

[3] Learning agents are agents that have a learning/optimization algorithm as their core algorithm. Given training data (a labeled dataset), those agents will attempt to increase their probability of picking the right label given some input data.

2.8 Natural Language Processing

2.8.1 Deterministic and stochastic grammars

[3] Deterministic grammars are grammars where the production rules have a probability of 1. Stochastic grammars are grammars where the probability of a given production rule may or may not be 1. What this means is that there may be various production rules for the same input tokens, but that each will have its own probability of being "used".

2.8.2 N-grams and HMMs

[4] N-grams are tuples composed of n elements. The elements might be either characters, words, or sentences. For instance, in the sentence "The quick brown fox jumps over the lazy dog", examples of character n-grams may be: (2-gram) th, he, e , q, qu, ui, ck, (3-gram) the, he , qu, qui, uic, ick; examples of words n-grams may be: (2-gram) the quick, quick brown, brown fox, (3-gram) the quick brown, quick brown fox, brown fox jumps.

[2] Through the use of n-grams, it is possible to construct a transition graph. If we count the number of cases where we transition from one n-gram state to another n-gram state, we can define a transition probability vector for all the subsequent n-gram state of a given n-gram state. We've effectively built an HMM.

2.8.3 Smoothing and backoff

[3] Smoothing is a simple technique where we assume that, even if we haven't seen any examples through a corpus, that at least 1 such example exists anyway.

[1] Backoff models are based on the idea that if not enough examples have been found in a corpus, we may rely on a less "constraining" conditional probability. For example, if we have an n-gram model and there aren't enough samples for the n-gram conditional probability, we might backoff to the (n-1)-gram probability.

2.8.4 Information retrieval

2.8.4.1 Precision and recall

[4] Precision is the number of relevant documents amongst the number of proposed/returned documents.

[4] Recall is the number of relevant documents amongst the number of relevant documents in the corpus. One can trivially achieve 100% recall by simply returning all documents.

2.9 Advanced Machine Learning

2.9.1 Definition and examples of broad variety of machine learning tasks

[2]
Generic

- Classification
- Regression
- Reinforcement learning

Specific

- Handwriting recognition
- Speech recognition
- Speech generation
- Object recognition
- Automated driving
- Recommender systems

2.9.2 Supervised learning

2.9.2.1 Learning neural networks

[2] Learning in neural networks is generally done through backpropagation and the use of gradients. First, an example is fed to the network, which computes its prediction. Given the correct answer (label), the network then computes the error given a loss function. This error is then backpropagated through the various layers of the network in order to update the weight of each layer according to their impact/weight on the error.

2.9.3 Ensembles

[1] Ensembles are based on the principle that you can take a group of different models and average their result to obtain an overall better result (the idea of synergy).

2.9.4 Nearest-neighbor algorithms

[2] k-nearest neighbor is a supervised learning algorithm that trains a model to do samples classification. Once the model is trained, new samples are matched with the most similar samples that have been seen in the trained model. The new sample will be given the same class as the one most of its neighbors have.

2.9.5 Unsupervised learning and clustering

2.9.5.1 K-means

[1] k-means is a clustering algorithm which takes a set of samples and will attempt to construct k groups given the features of these samples.

2.9.6 Reinforcement learning

2.9.6.1 Exploration vs. exploitation trade-off

[3] In reinforcement learning, given infinite time, one will want to explore all potential states in order to discover the path with the highest reward. However, reinforcement learning often occurs in environments in which there is a limited amount of time to test different possibilities. In this case, it often makes more sense to prefer paths that will have proven to be valuable over those that are less valuable, which is what is called exploitation.

2.9.6.2 Markov decision processes

[1] A Markov decision process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathcal{S}$: a finite set of states
- $\mathcal{A}$: a finite set of actions
- $\mathcal{P}$: a transition probability matrix, the probability of transitioning from s to s' through action a
- $\mathcal{R}$: a reward function, the immediate reward given for transitioning from s to s' through action a
- γ: a discount factor
- A policy π is a distribution over actions given states

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- The state-value function $v_\pi(s)$ is the expected return starting from state s and following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

- The action-value function $q_\pi(s, a)$ is the expected return starting from state s and following action a

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

2.10 Perception and Computer Vision

2.10.1 Computer vision

2.10.1.1 Image acquisition, representation, processing and properties

[3] Image acquisition is generally done through a device (e.g., a web camera) that converts light into a electrical/digital signal.

Such signal is converted into a 2D tensor where you have the (x, y) coordinate associated with an (r, g, b) color, which represents a single pixel. It is also possible to represent the color in other spaces such as BGR, HSV/HSL or YPbPr.

Processing of image is generally done through the use of filters or specific operations on the tensor representation of the image (such as extracting the min/average/max values, normalizing).

2.10.1.2 Motion analysis

[2] To do motion analysis, one tries to build a vector field, which attempts to map an initial pixel location (on a previous frame) with its new position on a new frame. A simple analysis will compute the difference between the initial and following images and indicate all pixels where the intensity/color has changed. This does not indicate how the pixels have moved, but that they have changed, indicating movement (or change in lighting intensity or other factors such as noise capture, etc.).

A more complex analysis will attempt to map the initial pixel location with its new location. is to minimize the amount of pixels that have changed as well as the distance the pixels will have moved.

2.10.2 Audio and speech recognition

[1] Audio recognition goal is to recognize specific audio segments or similar sounding audio segments, given that they may be distorted due to the capturing device or due to the presence of noise during capture. Recognition is generally an attempt at producing a fingerprint of the signal and searching through a database of known signal for a match.

Speech recognition is a complex domain in which one attempts to automate the extraction of text from audio data.

2.10.3 Modularity in recognition

[2] Recognition is generally composed of multiple steps, which can be composed as a pipeline of modules. For instance, in speech recognition, one will first want to process the audio signal in order to extract a noise profile which will then be removed from the original signal in an attempt to reduce said noise. The signal may then be normalized so that quiet and loud sounds are both at the same volume. The system may then attempt to segment the audio signal into words, which will then be send for recognition in isolation. Afterward, another module may receive suggestions for each of the processed word and attempt to construct a grammatically correct sentence.

2.11 Networking and Communications

2.11.1 Routing and Forwarding

2.11.1.1 Routing versus forwarding