Learning Resources for SE_14 – Artificial Intelligence Basics

0. Meta-Data:

Document Version: 1.0

Authors: Frank Trollmann

Last Change Date: 10.01.2020

Changelog:

Version	Author	Comment
1.0	Frank Trollmann	Created initial document.

1. Document Description

This document describes learning resources for module SE_14 – Artificial Intelligence Basics. Artificial Intelligence as a research field is quite big and quite old. It contains several sub-topics which themselves represent fields of significant size. Thus, any learning resource you come across will often be only partial or will be colored by the views and experience of the author. While this is not necessarily a bad thing, it can make it hard to relate the content of different learning resources to each other, especially if they focus on specific fields of AI, rather than the big picture. This document describes a big picture as a framework into which to place the different learning resources and the different areas of artificial intelligence. For each area a set of learning resources is given to enable students to understand the respective area in more depth. Where available, the learning resources try to offer various levels of details and learning styles to enable you to pick the best one for you.

A few additional notes about this document:

- Learning resources are presented in tables. So, you can easily scan for them.
- Each learning resource has a key. Square brackets will be used to reference / cross reference learning resources.
- Learning resources are accompanied by a comment. These comments give information on the content and level of the learning resource. They also tell you which parts of the resource (chapter, specific videos) to focus on. Reading them will make your life easier!
- Underlined fonts are used to highlight important statements.
- This document aims to tie together the learning resources. It does not aim to be a comprehensive description of the content of the learning resources. This means it will not be possible to pass the module without also going into the learning resources
- For Information in which knowledge you need for which level please refer to the module description.

The following sections give you the basics of artificial intelligence. We will use the concept of intelligent agent to conceptually tie the different areas of artificial intelligence together (cf. Section 2). Based on this generalized view we will cover Optimization (cf. Section 3), Machine Learning (cf. Section 4), Planning (cf. Section 5) and Reasoning (cf. Section 6). The last section (cf. Section 7) discusses other fields that are related to artificial intelligence. These fields also indicate other modules that may be combinable with this module in a project.

2. Artificial Intelligence = Intelligent Agent?

The first question we should tackle when talking about artificial intelligences (AI) is "What is an artificial intelligence?". This question is (not surprisingly) hard to answer.

The main problem is, that the question "What is intelligence?" is already hard to answer, as it is not a quantity that is directly measurable but is usually tied to certain types of behavior that we have come to associate with intelligence (e.g., solving problems, applying knowledge in creative ways, being able to acquire new knowledge). This is also how the intelligence test "measures" your IQ. It tasks and scores you based on how well you solve them. And that is why you will have different scores for different IQ tests you take. Because it's an approximation of an abstract concept and it simplifies a variety of different types of tasks into a single number.

On a similarly abstract level, we can say that the goal of artificial intelligence is to create an artificial being that behaves in a way that we would expect from someone with intelligence. There are a few different definitions of artificial intelligence in literature. For a more in-depth summary you can refer to the introduction of [RusselNorvig], where four schools of thought for defining artificial intelligence are classified. These can be defined as a parametrized statement like

"An AI is an artificial system that [thinks / acts] [rationally / like a human]"

The first of the dimensions here is thinking vs. acting, depending on whether the actions or the thought process of the artificial system is in focus. The second dimension describes whether the goal is to act rationally or to act like a human would (including deviations from absolute rationality due to traits like personality, emotions or impulsiveness). For example, the well-known Turing Test, which is supposed to detect an artificial intelligence, tests whether the artificial system acts like a human. Specifically, it tests whether a human is able to detect the difference between a real human and an artificial system posing as one during textual interaction (i.e., chatting).

For our discussions here we will use the definition "An AI is an artificial system that acts rationally", following the definition also used by [RusselNorvig]. This decision is somewhat arbitrary and has no claim to be the perfect one. For didactic reasons, it happens to be the definition that is best suited to tie together the content of this learning resource, as we can focus on actions (which are usually a lot easier to define in a computer system than thinking) and rationality (which can be measured by comparing actions to goals, which is easier than comparison to a real human being).

More specifically, we will follow [RusselNorvig] and define an artificial intelligence as a rational agent (cf. Fig. 1). An agent is a system that autonomously acts within an environment (e.g., a self-driving car that can drive around in the streets of a city). To perceive information about the environment, the agent has a set of *sensors* (e.g., the RADAR or LIDAR sensor or camera-based sensors of a self-driving car). To interact with the environment, the agent has a set of *actuators* (e.g., acceleration, steering position, turn signal). To define when an agent is rational, we need to define what rationality is. To do so, we need an explicit *goal* representation (sometimes also called performance metrics or performance measure). The goal could be a condition, the agent wants to fulfill (e.g., being at the destination position), a situation the agent wants to avoid (e.g., a collision) or a function the agent wants to optimize (e.g., minimizing the fuel consumption or time to arrive at the goal). And we say that the behavior of the agent (i.e., the actions, executed by the actuators) is rational, if it is directed at fulfilling its goals.

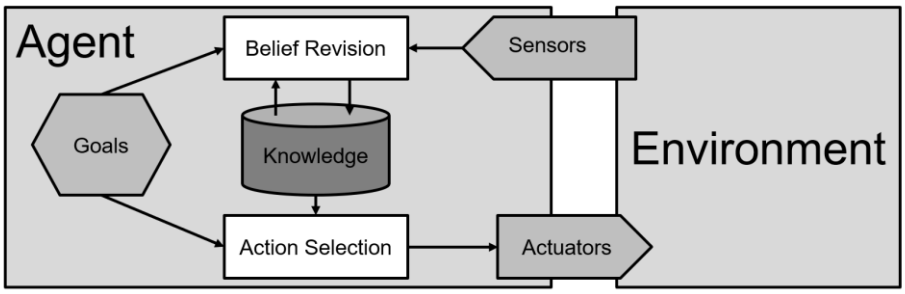

Figure 1: Illustration of the components of a rational agent

There are a lot of different approaches for implementing rational agents and the fields of autonomous systems and agent-oriented design are research fields of their own. For our purposes, we just use the concept as structuring mechanism, as it enables us to place the fields commonly associated with AI into the context of the agent. For this, we assume that the agent has an inner knowledge base that contains information about the environment and itself. With respect to this knowledge base we distinguish two major parts of the implementation of an agent:

- Belief Revision is the process of deriving information from sensors and adding it to the knowledge base. Since knowledge derived from sensors may be inaccurate or partial, this process may also require the correction of statements in the knowledge base or even the resolution of conflicts between existing knowledge and things perceived by the sensor.
- Action Selection is the process of determining which actuator to activate at which point in time. The process is usually based on the existing knowledge and the goals to achieve. In the following we will call the execution of an actuator an action.

Today, a lot of the field of AI is not directly concerned with how to implement a rational agent. The field now also consists of several sub-fields, like machine learning or planning, that represent technologies that can be used to implement an AI but would not constitute an actual AI, even if implemented perfectly. The module AI Basics focuses on four of these fields: Optimization, Machine Learning, Planning and Reasoning.

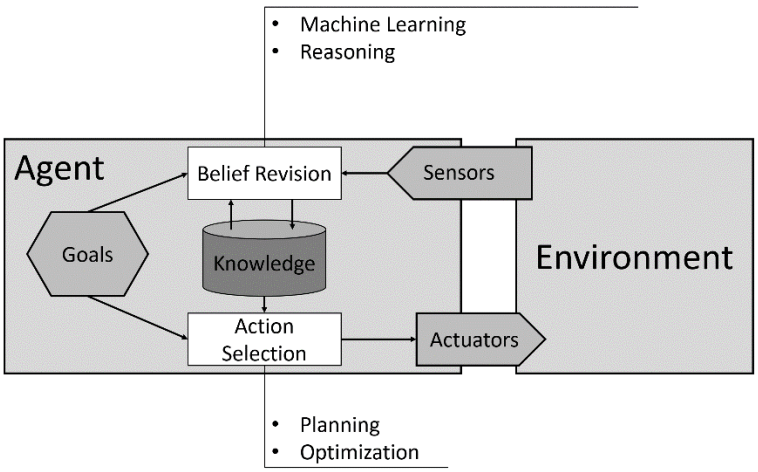

Figure 2: Optimization, Machine Learning, Planning and Reasoning in context of a rational agent

These can be placed into the definition of a rational agent as depicted in Figure 2. Machine Learning and Reasoning can both be used for belief revision. Both can be used to derive new knowledge.

Reasoning uses logical deduction to do this. E.g., by knowing that Waldi is a dog, we can derive that Waldi can bark. Machine learning derives new knowledge from statistical correlations. E.g., after repeatedly observing that pressing a button coincides with a light bulb switching on, it may be derived that the button switches on the light.

Planning and Optimization are both mechanisms that can be used to select actions. While Planning tries to derive a sequence of actions that will transition the agent to a goal state, optimization aims to find a configuration of variable assignments that optimizes a function (setting the variables to the assigned values is the action in this context).

These four techniques form the main content of the module Artificial Intelligence Basics and will be handled in more detail in Sections 3 to 6.

In general, the purpose of applying these techniques in a project will often not be to implement an artificial intelligence, but to solve an AI problem, a task that is usually associated with techniques from AI. This is another mechnism we will use to tie together the fields of AI in this learning resource. We will use an abstract problem definition for this purpose:

"I need an algorithm a that produces *o*, if I give it *i*. *o* should be *s*"

This problem definition needs an algorithm (a) (i.e., a specific algorithm you select / implement from one of the fields of AI) that, given a certain input (i) produces a specific output (o) that satisfies certain success criteria (s). The parameters *i*, *o*, and *s* change, depending on which area of AI the problem belongs to. This also implies that formulating such a problem for a task in your project should indicate to you, which area of AI is useful for solving this problem. The specific problem formulations for each area of AI will also be discussed in Sections 3 to 6.

To get more background information on intelligent agents and how they select actions you can use the following learning resources:

Resource	**(Recommended) Intro to Artificial Intelligence**
Key	IntroToAI
Type	Online Course, Video Lectures, Quizzed
Link	https://classroom.udacity.com/courses/cs271
Cost	Free
Comment	This is a great course that touches on many topics, even some that go beyond this module. If you are really interested in AI and all related topics, I recommend going through it. Some parts are a bit lighter touch (robotics, computer vision, natural language processing) but still informative. If you want a light touch overview first, this course may be a bit too much in detail for you. Here are some lessons on which you could focus in that case: • Lesson 1: Agents • Lesson 2/3: Problem Solving (planning) • Lesson 7 / 8 / 14: Machine Learning ○ These are already way more in-depth than you need for getting an overview. I recommend looking into the first parts of each lesson (Supervised, unsupervised, reinforcement) and maybe checking out two to three machine learning methods you are interested in. • Lesson 10: Basic Logics

Resource	**Artificial Intelligence: A Modern Approach**
Key	RusselNorvig
Type	Book
Link	https://www.amazon.de/Artificial-Intelligence-Modern-Approach-Global/dp/1292153962/ref=sr_1_1?keywords=k%C3%BCnstliche+intelligenz%3A+eine+modeln+approach&qid=1554123526&s=gateway&sr=8-1 Available in Library (SE96)
Cost	~ 50 €
Comment	This is the standard textbook on Artificial Intelligence. It motivates Artificial Intelligence from the point of view of a rational agent. It is also a good resource, especially to understand planning and reasoning.

Resource	**Video-summary of Artificial Intelligence: A Modern approach**
Key	AgentVideo
Type	Video
Link	https://www.youtube.com/watch?v=UjQ1AzSvCp8
Cost	Free
Comment	This YouTube video is a summary of chapter 2 of [RusselNorvig]. It contains the main content and a few more examples than the book.

Resource	**Artificial Intelligence Basics Slides**
Key	AIBasicsSlides
Type	Slides with Explanation text
Link	https://drive.google.com/drive/u/0/folders/1hMcpHHU0-OPlPcDI7mmax9-YC1z5xmma
Cost	Free
Comment	Slides from a learning unit on Artificial Intelligence. The learning unit was supposed to be an overview that touches on all the important AI topics and gives you terminology to search on your own. Please note that the "handout" version of the slides contains accompanying texts that is aimed at making the slide understandable without presenter. The content of this meta-learning resource is based on the content from that learning unit.

3. Optimization

Optimization aims to optimize a function. Specifically, it aims to find those input parameters, that optimize the output value of a function.

Let's define a simplified running example to illustrate this. Our function is the output of a production facility (e.g., measured in profit from selling the produced items, minus production costs). This output depends on the management of that facility. Let's assume that there are two parameters that can be changed by management: the number of workers employed at the facility (*#workers*) and the number of hours each worker works per day (*#hours*). The goal of optimization is then to find values for #workers and #hours that maximize the profit. Written down as the signature of a function, this

situation looks like this: *revenue: Integer x Integer → Double*. This means, the function takes two integers as input (#workers, #hours) and produces a double (the revenue).

Formulated as an AI problem, this looks like this:

"I need an algorithm *a* that produces *a parameter assignment* for this *function*. The parameter assignment should *maximize / minimize* the function."

The three parts of our general AI problem definition are defined as follows. The input is defined by a function. In our example, the revenue function that has two input parameters #workers and #hours. The output is an assignment of the input parameters of this function. In our example, the assignment consists of specific values for #workers and #hours. The success criteria is that the variable assignment optimizes the value of the function. Both maximization (finding the largest possible value) and minimization (finding the smallest one) are possible here. Indeed, both are somewhat interchangeable, as a maximization problem can be turned into a minimization problem by multiplying the function with -1.

There are different ways to solve optimization problems. If the function is known in mathematical notation and is differentiable, it is possible to use mathematical approaches to find its maximal or minimal values. This can be done by calculating the first derivation and finding those points where this derivation is zero (because this signifies a maximum, minimum or saddle point of the original function). Making use of this, it is often possible to directly calculate the solution to an optimization problem, especially for polynomials.

However, often the mathematical representation of the function is not known or not differentiable. This is the case in our revenue example, where we do not have a mathematical way to represent productivity of an individual based on working hours and thus could not write down a formula that calculates the revenue. In these scenarios, the usual strategy is to try out a few combinations of parameters and select the one with the best result.

In theory, once you try out all possible combinations of parameters, this also leads to the absolute maximum / minimum. However, in praxis, this is often not possible due to time or resource constraints, or because there simply are infinitely many parameters. In our revenue example, let's say trying out a combination of number of employees and working hours requires one month, to get a statistically significant revenue value. This means, even if we have only ten different numbers of employees and five different values for working hours, it'll take 50 months (i.e., more than 4 years) to find the optimum. This is often called the state space explosion: the number of states that needs to be searched to know for certain that you have found the optimum quickly exceeds the number of states that can be searched with the available resources.

Thus, most non-mathematical optimization algorithms aim at finding ways to determine which states should be visited in which order to maximize the likelihood of finding an optimal solution (or a solution that is close to being optimal). They make use of different properties of the search space. For example, it may make sense to start at one point and always go to the neighbor with the highest value, thus climbing higher and higher until you find a point where all neighbors are lower (this is called hill-climbing).

One thing that should be stressed is that there are different types of optimization problems. They differ in the types of parameters and properties of the function. In general, we can distinguish four types of parameters[1]:

- Categorical: These are parameters that can have a discrete amount of unordered values. An example are Color names (e.g., pink, red, blue, green, cyan). There is a finite number of named colors, but they have no inherent order.
- Discrete Numerical: These are parameters that have numerically ordered values of discrete nature. Examples are natural or whole numbers.
- Continuous Numerical: These are parameters that have numerically ordered values of continuous nature. An example are real numbers.
- Ordinal: These are categorical variables that nevertheless have a numerical order. An example are star ratings (one star, two stars, three stars, …) that represent a discrete set of categories, but has a numerical relation (two stars > one star).

Typically, Optimization is concerned with numerical values. However, in some cases the other values also can appear (beyond being border cases for optimization, they will also be relevant for machine learning, which is why they are listed here).

The applicability of optimization methods often depends on the type of input and output values of the optimized function. For example, hill climbing, as described above, is only possible if a notion of neighboring state exists. In our example of a revenue function, that has two discrete input values #workers and #hours, it would be applicable. But in other functions with continuous input parameters it could not be applied directly (although a generalization of hill climbing to continuous spaces exists and is called gradient ascent/gradient descent).

Another thing that influences the choice of algorithm is the form of the function. Is it guaranteed to have exactly one optimum, or are there multiples? Is it steadily growing towards the optimum or randomly jumping around in state space? As one can imagine, depending on these properties different approaches to finding the optimal value make sense.

So far, we've avoided talking about specific algorithms as much as possible. That's what the learning resources below are for, as depending on your problem you'll have to look into different algorithms. Here are a few general ones that you should know about, though: Hill Climbing, Gradient Descent, Simulated Annealing, Evolutionary Algorithms.

Resource	**Introduction to Optimization**
Key	IntroOptimization
Type	Online Videos
Link	https://www.youtube.com/playlist?list=PLLK3oSbvdxFdF67yVxF_1FQO9SbBY3yTL
Cost	Free
Comment	This playlist teaches you the basics of optimization and names a few algorithms to look into. If you have never heard of optimization before I recommend this series to get familiar with the overall problem and terminology and to get an intuition of how some of the existing optimization algorithms behave.

[1] Please note that there are different ways to categorize variable / measurement scales and this is just one of them. Especially Discrete and Continuous Numerical values are often just called numerical.

Resource	Gradient Descent
Key	GradientDescent
Type	Wiki
Link	http://wiki.fast.ai/index.php/Gradient_Descent#Gradient_Descent
Cost	Free
Comment	Good description of Gradient Descent, one of the first optimization algorithms you'll hear mentioned. Contains mathematics as well as example code. Also discusses some of the extensions of gradient descent.

Resource	Simulated Annealing
Key	SimulatedAnnealing
Type	Online Article
Link	http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6
Cost	Free
Comment	A description of the simulated annealing algorithm. Complemented by Example Code.

Resource	Multi-Objective Optimization
Key	Multi-Objective
Type	Online Videos
Link	https://www.youtube.com/watch?v=56JOMkPvoKs&list=PLwWiU_ClpuYpK0zDHR04J5pZA_Jh10uzE&index=2
Cost	Free
Comment	This video compares single- and multi-objective optimization. It gives a good intuition on the problems that arise when dealing with more than one goal function.

Resource	Evolutionary Algorithms
Key	Evolutionary
Type	Online Videos
Link	https://www.youtube.com/watch?v=L--IxUH4fac
Cost	Free
Comment	This video gives you an understanding of evolutionary algorithms but doesn't go into too much detail on how the respective parts of the algorithm are implemented.

Resource	Particle Swarm Optimization
Key	Particle
Type	Article
Link	http://www.swarmintelligence.org/tutorials.php
Cost	Free
Comment	this site gives a good overview of particle swarm algorithms, including the math behind them and pseudo code. It also points you to a few more detailed sites.

4. Machine Learning

Machine Learning as a field is concerned with learning correlations in structured data. For this, it applies statistical tools, to identify these correlations. The basic idea: machine learning finds patterns in data. There are a few different types of machine learning. However, before we briefly describe these, let's start with defining the AI problem for one of them: supervised learning.

Supervised Learning aims to learn a function from examples of its input and output values. For an example, we can use the revenue function we already used for optimization:

$$\text{revenue: Integer x Integer} \rightarrow \text{Double}$$

While optimization is concerned with finding the combination of input parameters (#workers, #hours) that maximizes this function, machine learning is concerned with finding a mathematical representation of this function. Or at least a mathematical representation of a function very close to it. To finds this function, supervised learning needs examples of input and output of the function. If our example production facility has already run for a few years, it may have experimented with different numbers of employees and working hours and may have recorded the revenues. These are examples that can be used for machine learning. They could look like this:

(5 Employees, 10 hours, 1.500 €), (7 Employees, 5 hours, 2.100 €) (8 Employees, 5 hours, 2.400 €) (10 Employees, 5 hours, 2.000 €), (9 Employees, 5 hours, 2.200 €), (11 Employees, 5 hours,1.800 €) (10 Employees, 8 hours, 800 €),…

Supervised Learning tries to find a function that would produce these data points, if applied to the input. For that it needs to make certain assumptions about the function (e.g., is it linear, is it a polynom, etc.). These assumptions are usually called a machine learning model. An example is linear regression, which assumes the function is linear.

An AI problem that requires (supervised) Machine Learning is usually defined as follows:

"I need an algorithm a that produces a function, if I give it a set of data points. The output of the function should be as close as possible to the output part of the data points."

Going back to our example, the dots in Figure 3 represent all data points that result from setting the number of hours to five (because 2 dimensional figures are much more readable). Applying a linear regression means, assuming there is a function of the form f(x) = a*x + b that explains these values. The values a and b are learned parameters. They are learned by a machine learning algorithm to minimize the distance between the function and the data points. Such a process could, for example, come up with the line shown in Figure 3. It is not possible to have zero distance to all data points, because the data points are not on a line.

This also illustrates that the choice of machine learning model matters a lot. In general, the more complex the model, the better it can cover data points. However, often, covering the data points with zero error is not desirable. The reason is that data points can be inaccurate. E.g., in our example, the revenue is not solely determined by the number of employees and working hours. It may also be determined by external factors such as the season (e.g., Christmas sales increase). We say the data contains noise: disturbances that are not caused by the function we are trying to learn and that must be factored out. Finding a model with zero error for all data points usually means we have learned the disturbances alongside the function. We say that the learning process has "overfit" to the data.

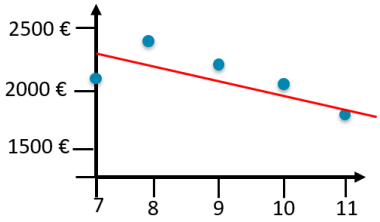

Figure 3: revenue function (fixing #hours to 5)

Linear regression is just one of many machine learning models. As with optimization, we'll not go into details of these models, but rather discuss factors that influence, which model you want to choose. And as with optimization, the type of your variables is a major factor. Depending on whether you are dealing with categorical, numerical or ordinal variables, different machine learning algorithms are used. In fact, two of the main areas of machine learning are defined as supervised learning with a different function type:

- Classification: is a supervised learning problem with a function of categorical value.
- Regression: is a supervised learning problem with a function that of (continuous) numeric value.

Functions with discrete numeric or ordinal value can usually be tackled with classification or regression approaches, or a combination of them.

The opposite of supervised learning is unsupervised learning. The difference is that the data does not contain an explicit output value for the function. In terms of a learned function we could say that we have no knowledge of the output value. Unsupervised learning approaches are learning the output value along with the function. This gives rise to a different area of machine learning:

- Clustering: is an unsupervised learning problem that has categorical output.

Clustering is useful when you want to categorize data points based on their spatial relation (e.g., learn a notion of "that blob of data points there that seems to be very similar"). While clustering will not tell you what these similarities mean, it is good at uncovering those similarities in the first place.

In between supervised and unsupervised learning approaches lies semi-supervised learning. Here, only some of the samples have output values and some don't.

Another big area of machine learning is called reinforcement learning. This is very closely related to the concept of agents and actions. The function that is learned is of the following form:

action_selection: Knowledge -> Action

It relates to some part of the knowledge of an agent and determines which action the agent takes in a certain situation. If this function was learned in a supervised manner, it would be trained with the right actions to take in each situation. In reinforcement learning the "right" action is not easy to determine. Instead, a reward function, which is often only possible to determine at a later point in time, is used to indicate which actions are better than others (because they led to a higher long term reward).

These four areas (supervised, unsupervised, semi-supervised and reinforcement learning) are the main areas of machine learning and depending on your AI problem you are usually working in one of them.

There are some other differences in algorithms. For example, how much data they can be trained on at a time, how much time is required for training, how many samples they need to provide reasonable results, etc. You can find some of these when researching algorithms to use.

One thing that should be pointed out explicitly is the relation between machine learning and optimization. (Supervised) machine learning aims to explain all samples by learning the right parameters of the respective machine learning model. This is expressed by defining an error function (e.g., the mean distance between the learned function and data samples). The learning process then is the task of finding those parameters for the machine learning model that minimize the error function. E.g., for a linear model that is described as the function f(x) = a*x + b, the machine learning process finds those values of a and b that minimize the error. If this sounds familiar, it is because this is an optimization problem. And indeed, optimization is what is often behind the learning process of machine learning. For example, backpropagation, the learning algorithm for neural networks, uses gradient descent, one of the optimization algorithms we mentioned above.

Resource	**(Recommended) Intro to Machine Learning (Udacity)**
Key	MachineLearningIntro
Type	Online Course
Link	https://classroom.udacity.com/courses/ud120
Cost	Free
Comment	Course is good and well-integrated. The target audience is a beginner and the course covers the general machine learning / data science layer, as well as the mathematics behind some individual algorithms. The course is very good but will also require you to invest some time! It does not handle Neural Networks.

Resource	**Model Buildling and Validation**
Key	ModelSelection
Type	Online Course
Link	https://classroom.udacity.com/courses/ud120
Cost	Free
Comment	This course is a good complement after you have familiarized yourself with a few machine learning methods. It is not about specific methods but about how to model your problem, how to select an appropriate model and how to validate that model.

Resource	**Intro to Artificial Intelligence Course**
Key	IntroAI_ML
Type	Online Course
Link	https://classroom.udacity.com/courses/cs271
Cost	Free
Comment	This course covers large parts of AI (see comment in the AI General Section). For Machine Learning I recommend looking into the following lessons: - Lesson 7: Machine Learning - Lesson 8: Unsupervised Learning - Lesson 14: Reinforcement Learning

Resource	**Machine Learning (Coursera & Stanford)**
Key	IntroML_Coursera
Type	Online Course
Link	https://www.coursera.org/learn/machine-learning
Cost	Free
Comment	This course is well-structured and well-explained. The audio quality is not super high at times, but it's still understandable. The course handles general machine learning and meta-learning concepts as well as math behind several important models. It is not always available. You can enter for free, but lectures are unlocked at specific times.

Resource	**Neural Networks Playlist by 3Blue1Brown**
Key	3B1B_NN
Type	Youtube Playlist
Link	https://www.coursera.org/learn/machine-learning
Cost	Free
Comment	This is a series of videos that lets you understand the mathematics behind neural network. I like this series quite a lot as it conveys a good intuition along with the mathematical descriptions. So even if you don't understand the math, you'll probably understand what they do.

Resource	**Python Machine Learning**
Key	Python_Machine_Learning
Type	Book
Link	ISBN: 978-1783555130 e.g.: https://www.amazon.de/Python-Machine-Learning-Sebastian-Raschka/dp/1783555130/ref=pd_sim_14_6?_encoding=UTF8&psc=1&refRID=SBREAZ8G2G65K5CPWNPG
Cost	40 € (in CODE Library, SE 12)
Comment	This book strikes a really good balance between theory and practice. While it sounds like it will mostly be concerned with the frequently used Python libraries for Machine Learning, it is very good at complementing the practical side in Python with mathematical knowledge about the underlying machine learning techniques. I highly recommend checking this book. Especially Chapter 1 gives a good overview of the field!

Resource	**Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems**
Key	Scikit-Learn
Type	Book
Link	ISBN: 978-1491962299 e.g.: https://www.amazon.de/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291/ref=pd_sim_14_4?_encoding=UTF8&psc=1&refRID=PHBEAB775FCDBZR0Z1MP
Cost	45 € (in CODE Library, SE 13)

Comment	The book is a good overview of Machine Learning methods and their implementation in the relevant Python Libraries. The introduction gives a good overview of machine learning in general. I especially liked the description of the no free lunch theorem on Page 30. The advantage of this book is that it enables you to understand meta-learning processes such as model selection and parameter tuning. The disadvantage is that it does not go into the mathematical foundations of the respective machine learning techniques as much. Similarly, data science processes are not handled in as much detail as they should be. So if you are interested in those aspects you'll need to find these details elsewhere. If you are taking AI basics, you can safely skip Section 11 to 15. They handle advanced techniques in Neural Networks, that already fit into SE_15 Advanced Machine Learning.

Resource	**The Elements of Statistical Learning**
Key	Statistical_Learning
Type	Book
Link	ISBN: 978-0387848570 e.g.: https://www.amazon.de/Elements-Statistical-Learning-Prediction-Statistics/dp/0387848576/
Cost	60 € (in CODE Library, SE 70)
Comment	Comment: This book describes a variety of machine learning methods from a mathematical perspective. This means, it is a good book to really understand how the respective methods work, but it requires some mathematical background knowledge, especially in statistics, to be understandable. If you have attended the Machine Learning Basics course and wanted to know more on the mathematical background behind the presented machine learning approaches, this book is a good complement. Be aware, that, due to the statistical background of the authors, the terminology is slightly different.

Resource	**Machine Learning: The New AI (MIT Press Essential Knowledge)**
Key	NewAI
Type	Book
Link	ISBN: 978-0262529518 e.g.: https://www.amazon.de/Elements-Statistical-Learning-Prediction-Statistics/dp/0387848576/
Cost	12 € (in CODE Library, SE 37)
Comment	The book covers machine learning from a purely conceptual perspective. Different machine learning algorithms are covered, but not discussed in depth. To me this is more a management summary of machine learning. It may be useful as an entry / overview but does not cover enough details to be used as a stand-alone learning resource.

5. Planning

Planning is closely related to the action selection of an agent. It answers the question "which sequence of actions will lead to the goal state". Examples of planning can be found throughout our daily life. For example, making a cup of tea requires us to boil water, then pour it into a can, add tea bags, wait a few minutes and then pour the tea into a cup. Each of these steps is an action (and we are the agent). Planning tells the agent in which order to apply which action in order to arrive at a goal state.

In general, planning requires the following parts:

- A notion of state. In our running example, the state is our knowledge about the world (is there water in the can? Is there water in the cup? Is the water hot? Are teabags in the can? etc.). In the context of an agent, the state is usually reflected in the knowledge of the agent.
- A set of actions that can be performed to change the state. In our running example, we have actions like "heat water","pour water into can" etc. In the context of an agent, the actions are given by the actions of the agent.
- A state-transition function that describes the effect of actions. This function lets us predict what the follow up state is, if we perform an action in a state. In the context of making tea, this function is implicit in our experience on the effect of actions (e.g., our knowledge about the water heater tells us that water will get hot if we switch it on, our knowledge about physics will let us predict what happens if we pour water). In the context of an agent, this is usually contained in the knowledge of the agent.
- A goal function that describes which states are desirable. This function lets us judge whether a state we have found is a valid result of the planning process. While making tea, our goal is to have a cup of tea in front of us. In context of an agent, the goal is encoded explicitly as the agent's goal function.

Given these components, a planning process is a function that produces a sequence of actions, such that these actions, when applied to the current state in the given order, lead to a state that fulfils the goal function. Accordingly, an AI problem that requires planning looks like this:

"I need an algorithm *a* that produces a sequence of actions, if I give it a start state, a goal, a set of actions and a state transition function. The result of applying the sequence of actions should lead to a state that is a goal state.

Or to say it in a simpler way: Planning finds a way from the current state to a goal state by applying actions. This is illustrated in Figure 4. The current state is the one on top of the image. From this state, we can apply three actions to arrive in the three mid-level states. From each of these states we can apply the three actions again, to arrive at other states (and so on). By applying Action 2, then Action 1, we can arrive at a goal state (marked in green). Planning is concerned with finding this sequence. Since the search space is a tree (or a graph, if the same state can be reached via different sequences of actions), planning is also sometimes called tree-search or graph search.

As with optimization, we have a problem that is easy to solve, provided all states can be visited in a reasonable amount of time. However, this is often not the case. As can be seen in our toy example already, the number of states in the lower layers grows exponentially with the depth of the search tree. And in realistically sized planning problems, it can quickly grow into a very large number. This represents another case of the state space explosion.

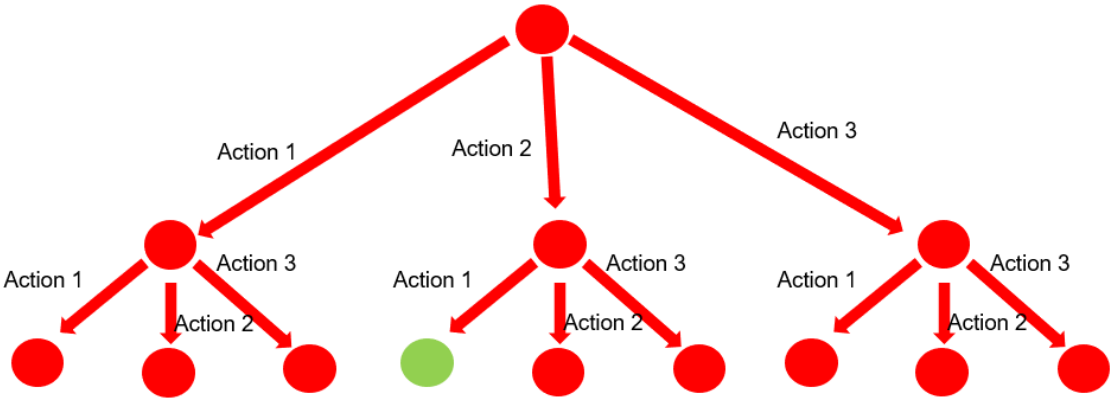

Figure 4: state space of a planning problem

To deal with this, planning algorithms aim to visit states in an order that has a high likelihood to find a goal state early. The most basic algorithms – depth-first and breadth-first search, prioritize going into the depth of one branch of the search tree (depth-first) or searching states in order of their distance to the start state (breadth first). These algorithms are often not enough to achieve good results as they don't take knowledge about the problem into account while searching for a goal state. More intelligent approaches utilize domain knowledge, to estimate which states are more likely to lead to a goal state. While planning for making tea, it may make sense to focus on those actions that happen in the kitchen and, e.g., prioritize actions like "turn on the television" way lower than "put teabag into cup". We call this estimation a heuristic. It tells the planning algorithm where in the search tree it is most likely to find a solution and thus directs the search.

It should be noted that the notion of planning problem we described here is the most basic notion. There are a lot of extensions of the planning problem. Examples are:

- Side-conditions like finding the shortest path to a goal instead of any path
- Actions with properties. For example, an action can be associated with monetary or time costs, could be applicable only in certain states (e.g., pouring out from a tea can, is only possible if the tea can is not empty).
- Uncertainty. There may be uncertainty in the knowledge that makes up the state or in the effect of an action (i.e., the state transition function).

Depending on these properties, different algorithms or different heuristics can be preferable. In the following learning resources, you can find some of them.

Resource	**(recommended) Intro to Artificial Intelligence Course**
Key	IntroAI_Planning
Type	Online Course, Video Lectures, Quizzes
Link	https://classroom.udacity.com/courses/cs271
Cost	Cost: free
Comment	This course covers large parts of AI (see comment in the AI General Section) and revisits planning several times along the way. The course uses slightly different terminology than we do above. It makes the distinction between problem solving and planning, where problem solving is what we called planning above and planning is when problem solving is interleaved with execution. For AI Basics we do not make this distinction and call everything planning.

	To learn about planning I recommend Looking into Lessons 2 and 11. Lessons 13 and 22 discuss additional complexity, like uncertainty or abstraction layers, that you only need to go into if you need them / are curious about them. Here's a short summary of these chapters content: • Lesson 2: Problem Solving (planning): You will learn about graph and tree search problems (i.e., planning problems), breadth first search, depth first search and A* with heuristics • Lesson 11: Planning: You will learn about interleaving planning and execution. You will also learn how to formulate a planning problem based on the knowledge representation of an agent. • Lesson 13: Planning under uncertainty: You will learn how to model uncertainty in the state transition function as Markov Decision Process and how to model uncertainty in knowledge caused by partial observability as Partial Observable Markov Decision Processes. • Lesson 22: Advanced Planning: You will learn advanced planning techniques, such as scheduling (planning in a time-based domain) and planning with abstract actions that themselves represent smaller plans (Hierarchical/Refinement Planning)

Resource	**MIT artificial Intelligence Course**
Key	MIT_Planning
Type	Video Lecture
Link	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/
Cost	Cost: free
Comment	This is a course about most areas of artificial intelligence, including planning. For planning, the following lectures are relevant: • Lecture 4: Lecture 4: Search: Depth-First, Hill Climbing, Beam Search. The lecture introduces the basic algorithms of planning: Depth first search and breadth first search. It also introduces some other algorithms (hill climbing and beam search) for planning problems that try to find a shortest path or a path with highest utility. Be aware that the hill climbing algorithm described here is not the same as the hill climbing algorithm you know from optimization. It shares similarities, but operates on a different state space (the search tree, rather than parameters of a function) • Lecture 5: Search: Optimal, Branch and Bound, A*: Describes more specific planning algorithms aimed at calculating shortest paths.

6. Reasoning

Reasoning is a technique that is closely related to the knowledge of an agent. It is concerned with the relation between individual pieces of knowledge. We call each piece of knowledge, a fact. Examples of relations between facts are:

- Mutual exclusion of negation: If I know a fact, the opposite cannot also be true. E.g., If my age is > 18 years, it cannot also be <= 18 years.
- Mutually exclusive facts: Only one of a group of facts can be true at a time. For example, my favourite colour is pink, so it cannot be blue, green, red, etc.
- Implication: If I know a fact, I can derive other facts from it. E.g., if I know that Waldi is a dog, it follows that Waldi can bark.

This means, individual pieces of knowledge can enable me to infer other pieces of knowledge. For example, if I know that you are a nineteen year old person that has a dog Waldi and the favourite colour black, then I also know that you are not underage, that pink is not your favourite colour and that your dog may bark.

In general, reasoning is the process of checking statements based on logical inference. Intuitively, this is a question answering of the type "is fact *f* true?". For example, I could ask "is your favourite colour red?" The answer to this question could be "yes", in case this can be inferred, "no" in case I can exclude red, because there is another favourite colour, or "don't know" in case neither one or the other applies. An AI problem associated to reasoning looks like this:

"I need an algorithm *a* that checks a fact, if I give it a knowledge base k and the fact *f*. The answer should be "yes", if *f* can be inferred from *k*, "no" if $\neg f$ can be inferred from *k* and "don't know" if neither *f* nor $\neg f$ can be inferred from *k*.

A reasoning algorithm can be applied for different purposes. It can be used to find out, whether a statement is known to be true (searching for answer "yes"). It can also be used to find out whether a fact conflicts with what I already know (searching for answer "no"). Another way to use it is to identify gaps in my knowledge (searching for answer "don't know") in order to fill them.

The naïve way to solve a reasoning problem is to infer all facts that can be inferred from the initial knowledge base, add them to that knowledge base and then check whether the fact *f* or it's negation $\neg f$ is among these facts. As you probably expect, this is usually not practical, as the amount of facts that can be inferred from a given knowledge base can quickly become very large or even infinite. For example, if my age is smaller than 18 years, it is also smaller than 19, 20, 21, … yeats. Accordingly, a reasoning algorithm doesn't try to infer all possible facts but tries to find a way to infer only *f* from the knowledge base. This may take multiple steps as it may need to infer a statement from which it can infer a statement from which it can infer a statement, (…) from which it can infer *f*.

Thus, the state space of a reasoning problem is a tree, starting with the initial knowledge base as root and containing all extensions of the knowledge base that can be generated by inferring new facts and adding them to the knowledge. This is illustrated in Figure 5. The initial state contains three facts: *color(a) = pink* (denoting that the favourite colour of *a* is pink), *of_age(a)=true* (denoting that *a* is of legal age) and *dog(b)=true* (denoting that *b* is a dog). Via the rules indicated in the example above, it is possible to derive statement $\neg$ *of_age(a)=true* as the opposite of *of_age*, statement $\neg$ *color(a) = red*, by exclusion, since the favourite colour is pink, and statement *can_bark(b) = true,* because we know that dogs can bark. Adding these to the knowledge base leads to the three next states. And from these states we can go on inferring more knowledge.

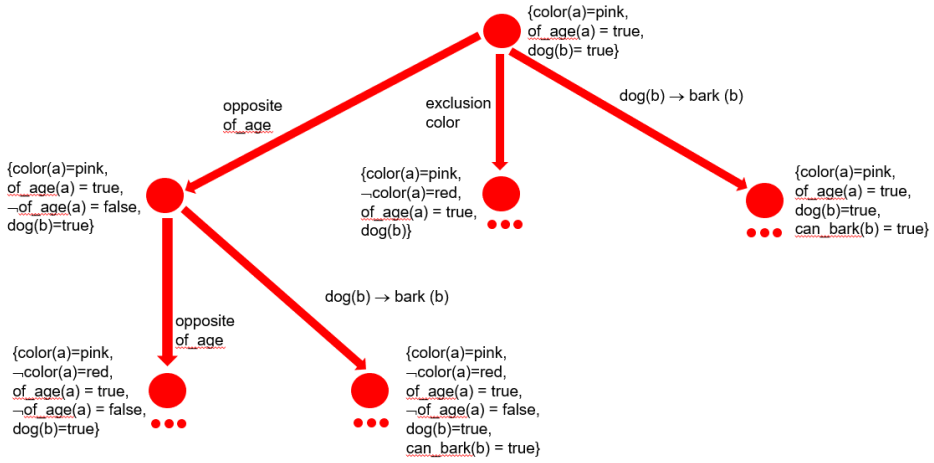

Figure 5: state space of a reasoning problem

The goal of the reasoning problem is to find a state that either contains the searched fact *f* or it's negation *f*. If this sounds familiar to you, then you've just discovered the close relation between reasoning and planning. Since reasoning spans a search tree, we can use planning algorithms to traverse this tree and find the goal state. So, it is possible to apply algorithms like depth first search or breadth first search to find a sequence of inferences to proof /disproof a statement. However, if we regard reasoning as one type of a planning problem, then there are also specific planning algorithms that can make better use of the properties of a reasoning problem.

Indeed, as with planning, machine learning and optimization, there are different variations of a reasoning problem as well. The two main differences are the knowledge representation and the inference system.

The knowledge representation is the language used to represent facts. There are several different formal logics that can be used as basis of reasoning. They differ in which kind of facts they can express, and which inferences they can support. The simplest and most widely used systems are propositional logics (enabling you to use negation, and, or, implication and equivalency) and first order predicate logics (additionally enabling the use of predicates and the use of existence and universal quantifiers). A lot of different extensions and variations of these logics exist for different purposes and domains. Examples are:

- Type Theory: Enables reasoning about types and their properties.
- Temporal Logic: Enables reasoning about the temporal order of events.
- Agent-based logics: Enables reasoning about agents, their beliefs, intentions and actions.
- Probabilistic logics: Enables the incorporation of uncertainty into facts.

As a rule of thumb, you should always try to use a logic that is as simple as possible, but enough to express what you need it to express. The reason is that any additional complexity in the logic system may either take away things you can reason about (because an underlying assumption has been relaxed) or provide additional complexity to the reasoning process.

The inference system is your source of inference. It usually consists of rules of the form $F1 \rightarrow F2$, meaning that if fact *F1* is known to be true then *F2* is also true. This means, if *F1* is in your knowledge base, then *F2* can be inferred. Inference rules can originate from different sources. Some of them originate from the underlying logic system you use. A simple example is $F1 \wedge F2 \rightarrow F1$, which means, if we know that *F1* and *F2* are both true then we also know that *F1* is true. You can usually find this type of inference rules for a given logical system by searching for "inference rules", "implications" or "provable identities". These inference rules represent the formal workings

of the used logical system. Our example of a mutual exclusion of negation (i.e., if I'm off age than I'm not underage) is such a rule. It can be expressed as $F \rightarrow \neg\neg F$ (i.e., if F is true then $\neg F$ cannot be true).

These inbuilt logical inferences only go so far, because they can only represent the workings of the underlying formalism and are agnostic when it comes to the meaning of statements. For example, our inference rules for barking dogs needs knowledge that there is a thing called dog and that it likes to bark. Any formalism that does not have explicit concepts for dogs and barking cannot express this rule as an inbuilt inference rule. This often makes it necessary to express custom inference rules that are specific to the domain or application currently used. For example, *dog(a)* $\rightarrow$ *bark(a)* is a statement in predicate logic that expresses our inference rule. These additional rules can be formulated once and then used like normal inference rules.

One thing that should be mentioned here is the relation to mathematical proofs. Often, what a mathematical proof does is showing that based on some assumptions, a certain statement holds. One way to do this is to use inference rules to show that you can arrive at the respective statement when you start out with the assumptions. Reasoning does something very similar, which is why the reasoning task described above is also sometimes called proofing.

Below are a few resources for diving deeper into the topic.

Resource	**Intro to Artificial Intelligence Course**
Key	IntroAI_Reasoning
Type	Online Course, Video Lectures, Quizzes
Link	https://classroom.udacity.com/courses/cs271
Cost	Cost: free
Comment	Yet again, this course makes for a good learning resource. Specifically, for reasoning, Lesson 10 is recommended. It gives you an overview of the basic knowledge representations (propositional logic, first order logic). The course does not cover reasoning algorithms specifically, although the problem solving lesson (Lesson 2) and the planning lesson (Lesson11) presents some basic planning methods that could be used for that purpose. It is recommended to complement this course with some of the below learning resources for a deeper dive.

Resource	**Course "Theory of Computation" by Harry Porter**
Key	HarryPorter
Type	Video Lecture
Link	http://web.cecs.pdx.edu/~harry/videos/
Cost	Cost: free
Comment	This course is a video lecture on theoretical informatics. While many parts are not relevant to reasoning, the part "First-Order Predicate Logic and Gödel's Incompleteness Theorem", specifically the following lectures go a bit deeper into predicate logics, and what it means for a statement to be true and provable - Lecture 54: First-Order Predicate Logic: An Overview - Lecture 55: Truth, Meaning and Proof - Lecture 56: True Statements and Provable Statements

	While the course is not specifically about reasoning, the process of proofing a statement by searching a way to infer it from given knowledge is a nice manual way to illustrate what reasoning algorithms do.

Resource	**MIT artificial Intelligence Course**
Key	MIT_Reasoning
Type	Video Lecture
Link	https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/
Cost	Cost: free
Comment	This is a course about most areas of artificial intelligence, including reasoning. Be warned, this course is approaching the reasoning topic from a completely different angle than we have described here. It does so using some very interesting examples (solving a mathematical equation and a program answering questions about itself), which is why this learning resource is here. It requires some effort to transfer into the framework described above though. The main thing you should be aware of is that the goal trees described in these lectures are the state space of the reasoning process (i.e., the tree of inferring more and more knowledge) and not the goal representation we have been talking about in agents. The lecture talks about these trees as if they are always present and known, whereas we've seen them as being built through a planning process. If you make that connection the course is well worth your time. For reasoning you want to look at the following Lectures: • Lecture 2: Reasoning: Goal Trees and Problem Solving. The lecture uses a mathematical problem as an example and tries to come up with a way to solve this problem programmatically. Turns out that this way is starting at a point and deriving new versions of the equation until finished. i.e., a planning/reasoning process. • Lecture 3: Reasoning: Goal Trees and Rule-Based Expert Systems. The lecture uses a few interesting examples to illustrate how questions can be answered in a rule-based fashion based on a goal tree. • Lecture 4: Lecture 4: Search: Depth-First, Hill Climbing, Beam. The lecture introduces the basics of planning and uses the reasoning example from Lecture 3 to illustrate them.

7. Related Research Areas and Modules

Artificial Intelligence involves a lot of techniques and has a lot of applications. Consequently, it is related to a lot of the modules offered at CODE. This section lists some of these relations, to indicate topics you may want to dive into and how they relate to AI.

The first thing that should be pointed out is that Artificial Intelligence Basics is a basics module. This means, it is designed to encompass a general understanding of all above mentioned techniques and their relation, but not to encompass deep dives into these topics. Right now, we have the module

SE_15 Advanced Machine Learning to encompass deep dives into machine learning topics. Planning, Reasoning and Optimization deep dives are not part of any modules as of now but would be good topics for the module SE_23_Speciality.

Two areas that are deeply related to artificial intelligence are data science and mathematics. Indeed, it is recommended to complement AI_Basics with the data science module, at least if your AI project is based on a dataset. Cleaning, visualizing and analysing a dataset are skills that are valuable in this kind of project and they are taught within the module SE_25_Data Science. In addition, you may want to take some of the math modules (SE_28 Linear Algebra, SE_29 Multivariate Calculus, SE_30, Probability and Statistics).

Finally, there are a few modules that represent potential applications of artificial intelligence. It may be the case that you implement a system based on images (e.g., a classification of pictures). In this case you may want to combine AI Basics, with SE_18 Image Processing. You may decide to use the techniques from artificial intelligence to implement an actual agent. Be it a robot or an autonomous software entity. In this context you may want to combine AI Basics with SE_13 Autonomous Systems.