1 What Is an Intelligent System?

1.1 Introduction

As we will work with intelligent systems and controls throughout this book, it might be rather beneficial to attempt answering the question posed in the title of this introductory chapter. Although a clear definition is still under debate, the word *intelligence* has commonly been used to express some association with the capacity to understand, the ability to reason with logic, and the facility to acquire and apply knowledge. From a broader perspective, an intelligent system utilizes these traits to create some level of self-awareness. For such a system, this type of cognition – however limited – is then utilized to interact with the system's environment. The objective of such interactions may involve altering the behavior of itself or its environment, hence performing some type of control task. When we combine intelligence with control, we are referring to a subset of control techniques that implements some of the intelligent system's capabilities to learn about the characteristics of its environment and to use this inference to achieve some predefined outcome.

Often, we find the term artificial intelligence (AI) associated with the capabilities listed above. The term *artificial* arises from specifically relying on man-made competences that can be categorized as intelligent. AI has the objective to mimic human behavior by incorporating human-like capabilities. This type of nonbiological intelligence is expressed through the use of mathematical rules. Some of the conceived AI systems belong to the field of machine learning (ML). ML comprises algorithms that attempt to learn mathematical rules automatically from acquired or observed data. It is found to be useful not only for methods and algorithms, but also in data analysis and analytics, in so-called big data, and in software tools. Another term that is linked with AI and ML is deep learning (DL). DL is generally considered a subfield of ML and to some degree also part of AI. DL particularly constitutes algorithms that are associated with multilayered neural networks.

Intelligent control techniques are especially convenient when dealing with incomplete or inadequate system representation. Their application also extends to situations where we deal with partial specification as well as uncertain environments. As there is missing information on both the target and the approach, learning becomes a key function in achieving the desired control objective. Hence, the learning component in such control systems emulates human intelligence. As we noted, a clear definition for intelligence is still deliberated upon, and so a clear definition of intelligent controls as a field is derived from and affected by this debate. An intelligent control system does leverage some of the advancements and algorithms developed for ML systems, such as neural networks, fuzzy inference systems, evolutionary algorithms, and combinations of any of these methods. In many applications of these methods the control design benefits from modeling a system based on data rather than on deriving governing equations utilizing physical laws, assumptions, and simplifications.

Intelligent systems are heavily dependent on computational resources. Hence, AI's development is rather tightly connected to any advancements achieved in computer technology. The early 1940s, when efforts were exerted to extract intelligence from coded communications during World War II, may serve as the origin of the development of modern computers. After World War II, in 1950, Alan Turing introduced the Turing test, which essentially provides for a means to determine if a computer can achieve the same level of thinking as a human. During the early 1950s computational statistics advanced to a level that allowed the inception of the field of ML. The generally accepted birthday of AI is traced back to 1956, when a small group of researchers attended the Dartmouth summer research project on AI. In particular, mathematics professor John McCarthy had proposed "to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [1]. The original inception of AI was followed by a number of scientific contributions in the subsequent years. For example, in 1965, Lotfi Zadeh expanded the infinite-logic field by introducing fuzzy logic [2]. The 1960s also saw major developments in the fields of natural language processing and computer vision. Those contributions were an ideal playground to make major advancements in robotics during the 1970s. For example, the first so-called intelligent robot, the WABOT 1 introduced in 1972, was able to engage in simple communications and perceive its environment by the use of artificial ears and eyes [3]. The original enthusiasm for AI and the ensuing expectations resulted in some disappointment as AI's capabilities were very much curtailed by the limitations of the available hardware. Intelligent systems and, in particular, AI have experienced repeated waves of attention followed by periods of disillusionment, where limited progress was achieved. Such epochs of disappointments and resulting loss of research efforts are termed *AI winters*.

The period from 1975 to 1980 has been termed the first AI winter, while the second AI winter occurred between 1987 and 1993. However, singular

achievements that caused public interest in this subject have repeatedly led to research activity and excitement in this field. For example, in 1997 an IBM supercomputer called Deep Blue competed against the world chess champion Garry Kasparov and defeated him in a six-game match [4]. Another such public exposure of advanced capabilities in intelligent systems that caused a revival of interest in AI was in 2015, when a neural network called AlphaGo beat the world's best Go player [5]. With the AlphaGo event the term *deep learning* was made popular beyond the research community. However, in between those times, intelligent systems invaded our homes in the form of smart appliances and device-based assistants. Today we are seeing AI capabilities embedded in all kinds of devices and systems. However, we are also dealing with some of the negative implications and challenges of intelligent systems. For example, we find ourselves debating liability and the legal ramifications of self-driving cars in the event of accidents, personal data and its analysis in consumer behavior, AI's role in bias and discrimination, and even legal personhood issues.

Although we introduce the underlying mathematical foundation of many of these intelligent system algorithms, a prominent topic throughout this book is the application of control systems using intelligent systems. Advancements in automation and in particular control systems have been a result of a shared and ceaseless quest by intelligent systems researchers and controls engineers to design machines that exhibit complete autonomy. Guided by a persistent belief that such levels of autonomy can be achieved by drawing expertise and knowledge from different fields, computer scientists, engineers, and mathematicians have been collaborating and contributing to this goal. Often, inspired by processes and behaviors found in nature, contributions are also made by researchers well outside of these disciplines. This broad alliance of disciplines is also reflected in the application of intelligent controls. We find intelligent controls algorithms nowadays applied to robotics, propulsion, communications, security, transportation, and logistics, to mention a few such applications.

1.2 Concepts in Machine Learning

Commonly, we categorize ML into three distinct groups: supervised learning, unsupervised learning, and reinforcement learning. When referring to supervision we imply having access to data that has labels. An example of labeled data is a repository of images depicting different leaves from different trees. Each picture itself is a data point, and its association with a tree is its label. Data with labels allow us to perform classification or solve regression problems. Classification will permit us to assign the proper label to a new image of a leaf. A regression problem is when we are seeking a model that lets us predict its output value. An example of a prediction problem is the regression model of forecasting house prices based on features such as number of bedrooms,

location, age, etc. The labeled data are the data available from the market (i.e., houses sold and their feature information).

Unsupervised learning handles data that does not have any association with labels. As there is no reference information or labels, unsupervised learning is used to perform clustering tasks, association analysis, and dimensionality reduction. Clustering is a function that authorizes us to group data into clusters. There is no label for the individual group or clusters, but the elements of each group share some distinct features. In ML we refer to a feature as a measurable characteristic. Features entail not only the characteristics of a data point, but also can be a descriptive attribute of the data point. In contrast to clustering, the association analysis refers to the process of finding prominent relations between features in a data set. An example of association analysis in practice is the process retailers perform on sales data to determine consumer behavior and preferences, which in turn can be used to drive promotion of certain products. The third possible application of unsupervised learning, the dimensionality reduction process, is a methodical approach to reduce the number of features in a data set. Generally, the more features a data set exhibits, the better an ML algorithm works for making distinctions between its data points. However, dealing with more features requires more computational resources. Hence, dimensionality reduction may help with the efficiency of an ML algorithm by supporting the elimination of features that contribute less than others to the clustering process.

Reinforcement learning (RL) refers to algorithms that employ a rewarding and penalizing mechanism to achieve a specific objective. The learning is subsidized by creating a reward when the algorithm moves in the right direction (i.e., approaches the stated goal). However, penalties are imposed when the opposite occurs. The rewards and penalties are a result of trials in the algorithm which is incrementally attempting to reach a stated goal. There are several types of RL methods: for example, the model-based and the model-free RL algorithms. The distinction is expressed by the type of implemented policy that the algorithm uses when deciding on rewards and penalties. When using model-based RL, the algorithm is interested in the underlying predictive model that defines the behavior of the system. This is achieved by considering the application of some type of intervention or input and then considering the results of the intervention, the next state of the system, and the immediate reward of the intervention. An example of model-based learning is dynamic programming. A model-free RL algorithm does not explicitly reference a model. Instead, it samples from experience or real observations from the environment rather than using predictions of the next state and next reward in order to change its behavior. An example of model-free RL is Monte Carlo control, Q-learning, or actor–critic learning algorithms. We will discuss those algorithms in more detail in Chapter 9.

Classification using supervised learning may entail methods such as the *K*-nearest neighbor, support vector machines (SVM), naïve Bayes classifier, decision trees, and random forests, to mention some of the more popular

algorithms. The *K*-nearest neighbor algorithm labels an unlabeled data point based on the majority label of its *K*-nearest neighbors. The number *K* (i.e., the number of neighbors) has great influence on the resulting class label: a small value of *K* allows for a greater influence of noise on the results; a larger number of neighbors, and hence a higher *K*, increases the computational cost in making the classification determination. Additionally, the choice of how we compute the distance to determine the neighbors is also a choice for the ML designer. Common distances utilized are the Hamming distance, the Manhattan distance, and the Minkowski distance. Support vector machines utilize a hyperplane to divide data points into classes. In order to evaluate the hyperplane parameters, and hence determine class association, perpendicular distances of the nearest data points close to the hyperplane are computed and maximized. This optimization problem is essentially a computational approach to maximize the margin between two classes. There are also multiclass SVM algorithms, which we will treat in more detail in Chapter 9. As the name indicates, the naïve Bayes classifier algorithm is a probabilistic classifier. This algorithm is an adaptation of Bayes' theorem to make prediction based on a data point's probability. The naïve reference indicates that the algorithm is indifferent to any dependencies of the different features used (i.e., we assume that all features applied in the algorithm are independent of each other).

Decision trees are simple algorithms using supervision to partition the data into smaller subsets. The partitioning – or branching – is a result of binary outcomes to a set of rules or questions. Essentially, each branch node of the tree embodies a choice among a number of possibilities and each leaf node represents a decision. Ultimately, the branching will cover all possible outcomes of a decision. The resulting structure of the flow chart resembles a tree; hence the name decision trees. An extension to decision trees is the random forests algorithm. A random forest algorithm is essentially a combination of multiple decision trees, where each tree is responsible for one unique class. The classification is determined by a majority voting computation among all the different classes and trees.

Unsupervised learning methods entail algorithms such as *K*-means, hierarchical clustering, and principal component analysis (PCA), to mention a few. The *K*-means clustering algorithm refers to a process where we determine partitions of a set of data points into *K* groups, which are called clusters. Although similar in principle to the *K*-nearest neighbor algorithm, it does not have access to data labels, and hence uses a similarity property of the data to form clusters. The discriminator utilized for finding similarities is a geometric property that is found by assessing the nearest mean or centroid. This point becomes a characterizing representative of the cluster. Another method for clustering is the hierarchical clustering algorithm. Beginning with considering the entire data set being assigned to individual clusters for each data point, the two nearest clusters are merged into a new cluster, eliminating the original two clusters. This process is

repeated until only a single cluster remains. There are different implementations of hierarchical clustering available, such as agglomerative hierarchical clustering and divisive hierarchical clustering. Principal component analysis is widely used in many disciplines, including in unsupervised ML. When we reference principal components we seek the most information of a data set by respecting the variance in the data. Here, the directions of the maximum information are the principal components. The process employs an orthogonal transformation to ensure that the variables of the process become uncorrelated.

1.3 Concepts in Deep Learning

Deep learning algorithms are closely associated with neural networks. The development of neural networks was inspired by observations of how the human brain functions. The human brain is literally composed of billions of neurons. A neuron is a nerve cell intertwined with other neurons and responsible for processing and transmitting chemical and electrical signals. From a biological perspective, dendrites are connections that obtain information from other neurons. Axons are used to transmit or send information, while a synapse is the connection between dendrites and axons. Processing of neural stimulation is handled by receiving multiple signals from different dendrites into the cell body. Once the accumulated signal meets or surpasses a threshold value, the cell body generates a signal that is conducted to other neurons by the corresponding axon. Since artificial neural networks are computer algorithms, we can use an analogy between the biological terms just introduced and the terms used to describe an artificial neural network. In this analogy, a biological neuron becomes an artificial neuron, while preserving the basic mathematical functioning of a biological neuron. A cell, also sometimes referred to as a soma, becomes a node, dendrites are inputs, and synapses become weights – which also function as interconnections. Finally, an axon is simply an output in the artificial neural network terminology.

The neuron in the artificial neural network not only contains the accumulation function but also houses a nonlinear activation function. This activation function supports the threshold operation, comparable to what the biological cell does. In an artificial neural network the inputs are weighted. The values of these weights are determined during the training – or learning – algorithm. Unlike humans, who have an astounding capability to learn using very few examples or data points, neural networks are much more in need of data and examples. For artificial neural networks backpropagation has shown to be one of the most effective learning algorithms. To apply this learning method we first process input data with the untrained network to generate an output, or prediction. This prediction is compared to the label associated with the input, also called ground truth. The comparison is the essence of supervision for training neural

networks, as the error is now fed back to the network to adjust the weights and biases of the network. The adjustment mechanism is essentially an optimization algorithm. For this optimization, we minimize an error, which is defined by a loss function. Popular loss functions in use are the mean square error or the cross-entropy loss functions.

A common optimization algorithm employed for minimizing the loss function is the stochastic gradient descent algorithm. When we feed the error back to the network, we compute the gradients and use this information along with a learning rate to adjust the weights and bias terms of the network. Since we use a lot of data and many examples to train such a network, we use specialized vocabulary to quantify learning iterations. For example, one *epoch* implies that we conducted one iteration using the entire data set. A *batch* is a subset of the entire data set. Hence, if we have 100,000 data points and a batch size of 2,000, then an epoch should contain 100,000/2,000 = 50 iterations.

Unlike in real life, for neural networks there is something like "too much learning." This is called *overfitting*, and occurs when we train a network for a data set to a degree that it only works for this single data set and not for other data sets, rendering the network useless. A network that is trained on one data set and can function well on another data set – representing similar characteristics – is a network that is *generalizable*. To monitor the training – or learning – process, we use a second set of data that is usually generated by splitting the original data set into two: a training set and a validation set. At instances during the training we determine the error of the network using the validation data set. At the moment the error based on the validation data set bottoms out, we assume that the network is trained, and any further training – even if the training data set provides lower subsequent errors – results in overfitting.

Up to this point we have described regular artificial neural networks. However, based on the section heading, we should wonder what DL is. The *deep* in deep neural network, and similarly in DL, is a reference to the number of layers such a network is constructed of. Regular networks have an input layer, a hidden layer, and an output layer. For such regular neural networks the described learning process works well. However, if we add more hidden layers the meaning of the error term used in the backpropagation algorithm is lost, and training becomes ineffective. Networks composed of multiple hidden layers are called deep neural networks. A concept called *dropout* – which is a form of regularization – and the use of different activation functions allow these deep neural networks to be trained in a similar fashion as regular neural networks. This inclusion in the learning process is referred to as deep learning.

With a multitude of layers, other operations can now be included into such networks. One such structure attempts to achieve greater autonomy and independence of human interaction by automatically extracting the features of the input data using operations such as convolution and pooling. We provide

a thorough review of these methods and computations in Chapter 8. Deep neural networks have found applications and provided solutions to problems that originate in face recognition, image classification, speech recognition, handwriting transcription, and medical diagnoses. Deep learning has been useful in decision-making, operation of equipment, including a range of different vehicles, social media, and – not to leave it out – the gaming industry.

1.4 Concepts in Intelligent Control

Control systems have been facilitating human activities for much of our species' existence. Seminal discoveries and inventions may have spurred new interests and areas of technologies utilizing controls or advancing controls itself. Nowadays, we find controls in every part of our lives, including in business, government, technology, and medicine, to mention a few. However, in this book, we deal with dynamical systems and its control in terms of the dynamical behavior of such systems. To describe the behavior we use mathematics, and hence the usage of control theory is dependent on employing this language. A driver for the application and the advancement of dynamic control system is the desire to reduce the necessary human work by increasing automation. Automation is dependent on controls engineering in order to perform automation tasks. The controller used in automation has the function of embedding some kind of strategic method and computation to achieve a specific outcome. Among control action outcomes, the stability of the system is one of the primary goals, especially if the system without control exhibits unstable behavior. However, the application of closed-loop control may also induce instability of the system. Hence, the study of stability is closely associated with control engineering. Many of the tools developed for these studies are dependent on precise system dynamic descriptions (i.e., mathematical equations characterizing the behavior of the closed-loop system). In some instances the acquisition of such descriptions is rather difficult or even impossible. For such situations, automatic inference of the system description through the control system has been used. These types of readings and extractions have found topical homes in system theories such as system identification, estimation, and adaptive control theory.

System identification is a process that collects input and output data from the operation of the system and uses these data to extract a system description, circumventing the utilization of physical principles to describe the system dynamics. Estimation theory makes ample use of statistics and allows for the estimation of system parameter values in stochastic environments. Adaptive control systems make use of estimation theory and system identification to adapt to changes in the system characteristics. As control systems depend on intervention (i.e., using actuators), the accounting of the exerted energy may be of interest, in particular

for optimizing effort and costs. Such considerations lead to the field of optimal control theory.

Other control system disciplines address issues in the robustness of the controller in the presence of system uncertainties. Dealing with exclusively nonlinear system behavior has resulted in theories encapsulated as nonlinear control systems. Specialization of specific control applications has resulted in a wealth of different specific control fields, such as reconfigurable control systems, resilient control systems, digital controls, and multivariable control systems, to mention a few. However, in this book we specifically look at intelligent control systems. The intelligent part in this field of study may function as an aid in evaluating controller parameters, defining and adjusting the controller structure, estimating the controller or system parameters, finding optimality, or even as a tool to define the entire system description.

Intelligent control systems is a broad field. In this book we focus on a few important concepts of intelligent control systems. In particular, we study fuzzy logic and how to utilize fuzzy logic theory to construct controllers. We use intelligent optimization methods such as genetic algorithms and particle swarm optimization to find optimal controller parameters, and we employ ML to infer models that represent the system to be controlled.

Considering a typical fuzzy logic control structure (Figure 1.1), the main components of such a system are the fuzzifier, the fuzzy knowledge base, a fuzzy rule base, an inference engine, and the defuzzification unit. The fuzzifier has the assignment to convert so-called crisp signals into fuzzy quantities. A crisp quantity is a reference to a signal that can have a numeric value – for example, a voltage reading of 3.309 volts is a crisp quantity. The fuzzifier associates this type of number into a degree of membership to a class. The class could be "high," "low," etc. and hence a degree of membership is expressed as some percentage of the voltage to be characterized as high and as low. The fuzzy rule base is responsible for modeling the knowledge of the system using fuzzy sets. For

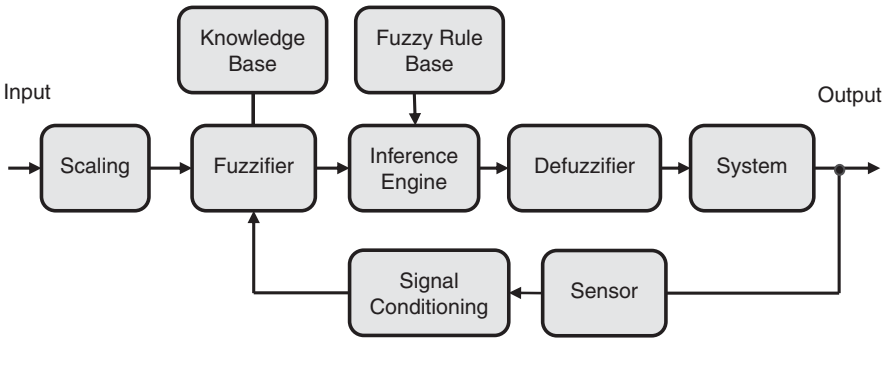

Figure 1.1 Fuzzy logic control system schematic.

example, the fuzzy rule base details the input and output relations using fuzzy relationship operators and membership functions. The inference engine embodies an abstract form of human decision-making by utilizing approximate reasoning, which is used to define the control strategy. Once the control action is decided upon, the decision – which is in the form of a fuzzy quantity – needs to be converted into a crisp output using the defuzzification process.

A system as depicted by Figure 1.1 can be hybridized with other intelligent control systems. For example, the fuzzification using membership functions to associate signals with linguistic variables can be optimized using heuristic methods such as tabu search, genetic algorithms, etc. A similar approach can be taken to find optimal rule sets for the inference engine. If simulations are required before the implementation, the system block is no longer a physical system, but a block representing the system dynamics. Such a block can be inferred using ML – for example, a trained neural network can be utilized to represent the physical system. Additionally, neural networks and fuzzy logic can be married to form neuro-fuzzy inference systems. An extension to such neuro-fuzzy inference systems is an adaptive neuro-fuzzy inference system where the inference system adapts to the changes of the system itself.

Extending the use of neural networks, and in particular deep neural networks, a different type of control system framework can be constructed: the RL controller or the reinforcement DL controllers (RL controllers). Such controllers represent rather powerful approaches to search-and-find-optimal controllers in the presence of nonlinearities, noise, and uncertainties. They can also be adaptive to system changes. RL controllers work sequentially by determining the impact of decisions made and control signals applied, which includes the consideration of future interactions. RL systems entail a reward function, which is used to evaluate and iteratively improve the policy, which determines the control action. A policy in RL is often given by a deep neural network, hence the learning component of the RL controller is responsible for optimizing the policy by adjusting the deep neural network using a cumulative assessment of the reward achieved.

1.5 Data, Signals, and Methods

Before indulging in specifics on controls and intelligent systems in the following chapters, we must recognize that all of these systems and approaches are driven by data. So, it is more than justifiable to contemplate the nature of data and how to classify it. We have already seen that data with labels allow the use of certain types of ML algorithms, and data without labels are open to being processed with other ML algorithms. Such differences can be utilized to further distinguish data sets into nominal versus ordinal data. Nominal data, from the Latin *nomen*, names or labels