Technical Elements of Machine Learning for Intellectual Property Law

Anthony Man-Cho So*

Abstract

Recent advances in artificial intelligence (AI) technologies have transformed our lives in profound ways. Indeed, AI has not only enabled machines to *see* (e.g., face recognition), *hear* (e.g., music retrieval), *speak* (e.g., speech synthesis), and *read* (e.g., text processing), but also, so it seems, given machines the ability to *think* (e.g., board game-playing) and *create* (e.g., artwork generation). This chapter introduces the key technical elements of machine learning (ML), which is a rapidly growing sub-field in AI and drives many of the aforementioned applications. The goal is to elucidate the ways human efforts are involved in the development of ML solutions, so as to facilitate legal discussions on intellectual property issues.

1 Introduction

Although the field of artificial intelligence (AI) has been around for more than 60 years, its widespread influence is a rather recent (within the past decade or so) phenomenon. From human face recognition to artificial face generation, from automated recommendations on online platforms to computer-aided diagnosis, from game-playing programs to self-driving cars, we have witnessed the transformative power of AI in our daily lives. As it turns out, machine learning (ML) techniques lie at the core of many of these innovations. ML is a sub-field of AI that is concerned with the automated detection of meaningful patterns in data and using the detected patterns for certain tasks.[1] Roughly speaking, the learning process involves an algorithm,[2] which

*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. All online materials were accessed on 30 March 2020. E–mail: manchoso@se.cuhk.edu.hk

[1] Shai Shalev-Shwartz and Shai Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press 2014).

[2] An *algorithm* is a well-defined sequence of computational steps for solving a problem. Specifically, it takes zero or more values as inputs and applies the sequence of steps to transform them into one or more outputs. Note that an algorithm can be described in, say, the English language (which is easier for humans to understand) or in a programming language (which is easier for the computer to process). The word *program* refers to an expression of an algorithm in a programming language. See Donald E Knuth, *The Art*

takes *training data* (representing past knowledge or experience) as input and outputs information that can be utilized by other algorithms to perform tasks such as prediction or decision making. With the huge amount of data generated on various online platforms,[3] the increasing power (in terms of both speed and memory) of computers, and advances in ML research, researchers and practitioners alike have been able to unleash the power of ML and contribute to the many impressive technologies we are using or experiencing today. In this chapter, I will give an overview of the key concepts and constructions in ML and, with an aim to make them more concrete, explain the roles they play in some of the contemporary applications. In addition, I will elucidate the ways human efforts are involved in the development of ML solutions, which I hope could facilitate the legal discussions on intellectual property issues. In recent years, there has been much interest in applying ML techniques to legal tasks such as legal prediction and classification of legal documents. However, the discussion of these applications is beyond the scope of this chapter.[4]

2 Main Types of Machine Learning

The outcome of any learning process depends, among other things, on the material from which the learner learns. As alluded to in the introduction, in the context of ML, the learning material comes in the form of training data. Since the training data in most applications of interest are too complex and too large for humans to process and reason about, the power of modern computers is harnessed to identify the patterns in and extract information from those data. A key characteristic of ML algorithms is that they can adapt to their training data. In particular, with better training data (in terms of volume and quality), these algorithms can produce outputs that have better performance for the tasks at hand. In order to distinguish among different ML tasks, it is common to classify them according to the nature of the training data and the learning process. In this section, I will describe three main types of ML tasks—namely *supervised learning*, *unsupervised learning*, and *reinforcement learning*—and explain how they manifest themselves in various real-life applications.

---

of Computer Programming. Volume I: Fundamental Algorithms (Third edition, Addison Wesley Longman 1997) for a more detailed discussion.

[3]The data could be in the form of images and texts posted on social media, browsing and purchasing history on e-commerce sites, or emails sent and received using online email platforms, just to name a few.

[4]Readers who are interested in some of the applications of ML in the legal field can refer to, e.g., Harry Surden, 'Machine Learning and Law' (2014) 89 Washington Law Review 87.

2.1 Supervised Learning

Supervised learning refers to the scenario in which the training data contain certain information (commonly referred to as the *label*) that is missing in the *test data* (i.e., data that have not been seen before), and the goal is to use the knowledge learned from the training data to predict the missing information in the test data. It has been successfully applied to various fields, such as credit risk assessment[5] and medical imaging.[6] To better understand the notion of supervised learning, let me highlight three of its key elements—preparation of training data, formulation of the learning task, and implementation of algorithmic solutions to perform the learning.

2.1.1 Preparation of Training Data

The word "supervised" in "supervised learning" comes from the fact that the training data contain information that guides, or supervises, the learning process. Typically, the information is supplied by humans (a process referred to as *labeling the data*). As such, it often requires substantial effort to prepare the training data for a supervised learning task.[7] To illustrate the concepts of training data and test data in the supervised learning setting, consider the task of recognizing handwritten digits. The training data can be a collection of handwritten digit samples, each of which is labeled with its interpretation (i.e., `0-9`). Figure 1 shows a small portion of such a collection from the MNIST database.[8] Any collection of handwritten digit samples that have not been labeled or seen before can then be the test data.

It is important to note that in general the label given to a data sample is not guaranteed to be correct. This can be caused, e.g., by human error or by the ambiguity in the data sample itself. For instance, in labeling handwritten digit samples, mistakes can occur when the handwritten digits are hardly legible (Figure 2). As the premise of supervised learning is to use the knowledge learned from the labels of the training data samples to predict

[5]Dinesh Bacham and Janet Yinqing Zhao, 'Machine Learning: Challenges, Lessons, and Opportunities in Credit Risk Modeling' (2017) 9 Moody's Analytics Risk Perspectives: Managing Disruption 30.

[6]Geert Litjens and others, 'A survey on deep learning in medical image analysis' (2017) 42 Medical Image Analysis 60.

[7]Nowadays, it is common to use crowdsourcing to get a large volume of data labeled. One manifestation of this is the use of CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) on various websites. Although the explicitly stated purpose of CAPTCHAs is to authenticate users as humans (to prove "I'm not a robot"), the responses given by human users provide information about the queries posed by CAPTCHAs (e.g., identify the traffic lights in the image, transcribe the distorted words, etc.), thus labeling the data in those queries in the process. See, e.g., Luis von Ahn and others, 'reCAPTCHA: Human-Based Character Recognition via Web Security Measures' (2008) 321(5895) Science 1465, for a discussion.

[8]Yann LeCun, Corinna Cortes, and Christopher JC Burges, 'The MNIST database of handwritten digits' (2010) ⟨http://yann.lecun.com/exdb/mnist/⟩.

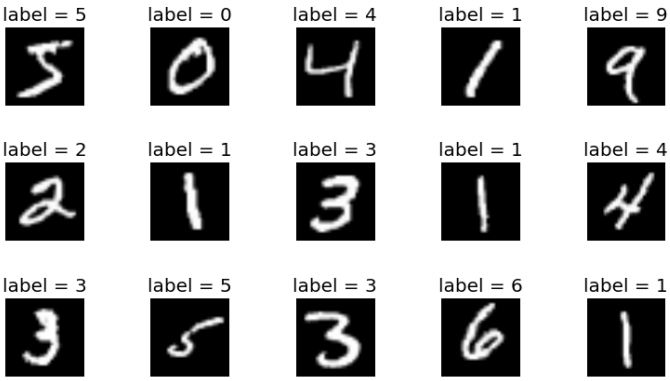


Figure 1: Sample handwritten digits from the MNIST database with their corresponding labels.

the labels of the test data samples, the presence of incorrectly labeled data samples can adversely affect the outcome of the learning process.

Figure 2: An ambiguous handwritten digit: Is this a '0' or '6'?

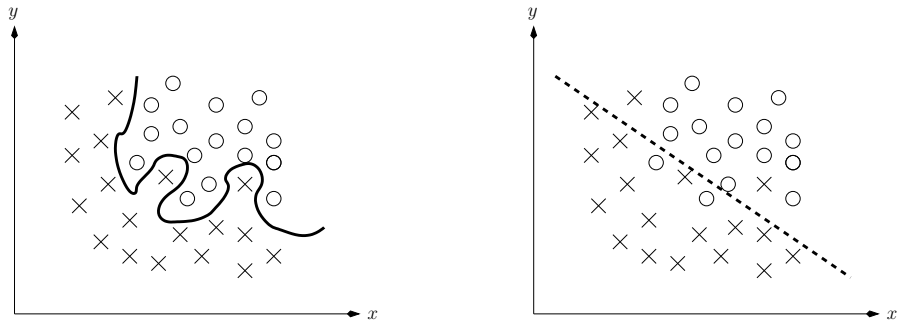
2.1.2 Formulation of Learning Task

The prediction of the labels of the data samples relies on a *prediction rule*—i.e., a function that takes a data sample as input and returns a label for that sample as output. With this abstraction, the goal of supervised learning can be understood as coming up with a prediction rule that can perform well on most data samples. Here, the performance is evaluated by a *loss function*, which measures the discrepancy between the label returned by the prediction rule and the actual label of the data sample. The choice of the loss function is largely dictated by the learning task at hand and is commonly known.[9]

To achieve the aforementioned goal, a natural idea is to search among rules that minimize the loss function on the *training data*. In other words, we aim to find the rule that best fits our past knowledge or experience. However, without restricting the type of rules to search from, such an idea can easily lead to rules that perform poorly on the unseen *test data*. This phenomenon is known as *over-fitting*. As an illustration, consider the task of classifying data points on the plane into two categories. Figure 3 shows the training data, in which each data point is labeled by either a cross "×"

[9]See, e.g., Shalev-Shwartz and Ben-David (n 1) for a discussion of different loss functions.

or a circle "$\circ$" to indicate the category it belongs to. A prediction rule takes the form of a boundary on the plane, so that given any point, the side of the boundary on which the point falls will yield its predicted category. Given a boundary, a common way to measure its performance on the training data is to count the number of points that it misclassified. Naturally, the fewer misclassified points, the better the boundary.



(a) Fitting by arbitrary curve (b) Fitting by line

Figure 3: Illustration of over-fitting in a classification task.

Suppose that we do not restrict the type of boundaries we can use. Then, a boundary that misclassifies the fewest training data samples is given by the bolded curve in Figure 3a. Indeed, all the crosses are on the left of the curve, while all the circles are on the right. However, such a boundary fits the training data too well and is not well-suited for dealing with potential variations in the test data. In particular, it is more likely to return a wrong classification for a test data sample.

On the other hand, suppose that we restrict ourselves to use only straight-line boundaries. Then, the dotted line in Figure 3b yields the best performance among all straight lines in terms of the number of misclassified training data points. Although the dotted line incorrectly classifies some of those points (e.g., there are two circles on the left and two crosses on the right of the line), it can better handle variations in the test data and is thus more preferred than the curved boundary in Figure 3a.

The above discussion highlights the necessity to choose the type of prediction rules that will be used to fit the training data. Such a choice depends on the learning task at hand and has to be made by human users before seeing the data. In general, there are many different types of prediction rules that one can choose from. Some examples include polynomial functions, decision trees, and neural networks of various architectures. A key characteristic of these different types of rules is that each type can be defined by a set of *parameters*. In other words, each choice of values for the parameters corresponds to one prediction rule of the prescribed type. For instance, in the classification example above, the straight-line boundaries

used in Figure 3b, which are lines on the plane, can be described by two parameters—slope and intercept. As another illustration, let us consider neural networks, which constitute one of the most powerful and popular types of prediction rules in ML today. Roughly speaking, a neural network consists of nodes (representing *neurons*) linked by arrows. Each arrow has a *weight* and connects the output of a node (i.e., the tail of the arrow) to the input of another node (i.e., the head of the arrow). Each node implements a function whose input is given by a weighted sum of the outputs of all the nodes linked to it, where the weights are obtained from the corresponding arrows. The *architecture* of a neural network is specified by its nodes, the links between the nodes, and the functions implemented on the nodes.[10] The weights on the links then constitute the parameters that describe different neural networks with the same architecture. Some commonly used neural network architectures include *autoencoders*, *convolutional neural networks* (CNNs), *feedforward networks*, and *recurrent neural networks* (RNNs). Each of these architectures is designed for particular learning tasks.[11]

Figure 4 shows an example of a simple three-layer feedforward neural network. It takes three inputs, which are denoted by x, y, z. The weight assigned to an arrow is given by the number next to it. All the nodes implement the same function, which is denoted by $f(\cdot)$.[12] To get a glimpse of what is being computed at the nodes, let us focus on the shaded node. It has two inputs, one from the output of the first node in the first layer, the second from the output of the second node in the first layer. The former, which equals $f(x)$, has a weight of 0.8; the latter, which equals $f(y)$, has a weight of 0.6. Therefore, the output of the shaded node is computed as $0.8 \times f(x) + 0.6 \times f(y)$. By assigning a different set of weights to the arrows, we obtain a different neural network with the same architecture. As an aside, one often sees the word "deep" being used to describe neural networks nowadays. Loosely speaking, it simply refers to a neural network with many (say, more than 2) layers.

Once the human user specifies the type of prediction rules to use, the next step is to find the values of the parameters that minimize the loss function on the training data. This gives rise to a computational problem commonly known as *loss minimization*. By solving this problem, one obtains as output a prediction rule of the prescribed type that best fits the training data. The rule can then be integrated into other decision support tools to inform the decisions of human users.

[10] Shalev-Shwartz and Ben-David (n 1).

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (http://www.deeplearningbook.org, MIT Press 2016).

[12] Mathematically, a function can be regarded as specifying an input-output relationship. The dot "$\cdot$" in the notation "$f(\cdot)$" represents a generic input to the function f. Given a number t as input, the function f returns the number $f(t)$ as output.

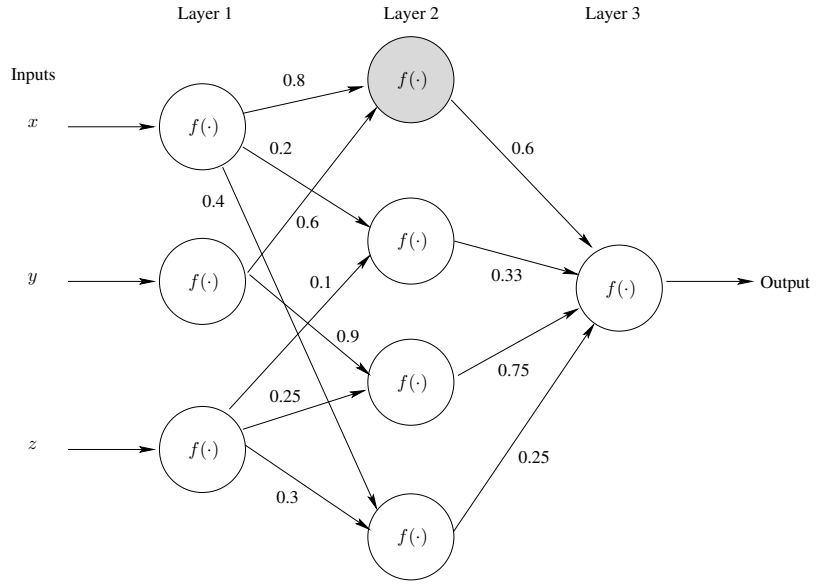


Figure 4: A simple feedforward neural network.

2.1.3 Implementation of Algorithmic Solution

Loss minimization problems are typically solved by *iterative algorithms*. Starting from an initial choice of values for the parameters, which can be viewed as a point in space, these algorithms proceed by moving the point in a certain direction by a certain distance, and then repeat until certain stopping criterion is met. Different algorithms have different rules for determining the direction and distance to use at each point and have different stopping criteria. Generally speaking, the directions and distances are designed in such a way that the values of the loss function evaluated at the points generated by the algorithm have a decreasing trend (recall that the goal is to minimize the loss function), and the algorithm stops when no further progress can be made. One popular iterative algorithm for solving loss minimization problems is the *stochastic gradient method*. At each step, the method moves the current point along a random direction that is generated based on the properties of the loss function, and the distance by which the point is moved is decreasing as the method progresses, so as to avoid overshooting the solution.[13]

Although algorithm design requires human efforts and it is natural for developers to protect their algorithms in some ways, the specifications (i.e., the rules for choosing directions and distances, and the stopping criterion) of many iterative algorithms used in the ML community are public knowledge. Still, even after one settles on a particular iterative algorithm to

[13]Sebastian Ruder, 'An overview of gradient descent optimization algorithms' (2016) ⟨https://arxiv.org/abs/1609.04747⟩.

solve the loss minimization problem at hand, the choice of initial values for the parameters (also known as the *initialization*) could affect the performance of the algorithm. To understand this phenomenon, let us consider the scenario shown in Figure 5. The points on the horizontal axis represent possible values of the parameter, and the curve represents the loss function $\mathcal{L}$. One can think of the curve representing $\mathcal{L}$ as a mountain range and an iterative algorithm as a person hiking there without a map and can only explore her immediate surroundings to decide on which way to go. The goal of loss minimization can then be understood as finding the lowest point on the mountain range. In Figure 5, this is the black dot corresponding to the parameter value w^* and loss function value $\mathcal{L}(w^*)$.

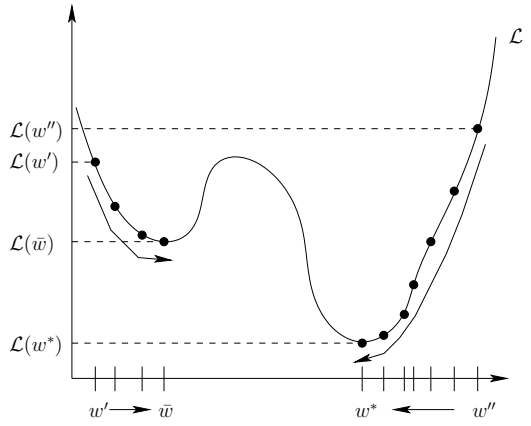


Figure 5: Effect of initialization.

Now, suppose that the hiker starts at the leftmost black dot on the mountain range. This corresponds to initializing the algorithm at the point w' whose loss function value is $\mathcal{L}(w')$. To get to a lower point on the mountain range, the person will naturally walk down the valley until she reaches the point with value $\mathcal{L}(\bar{w})$. At this point, the hiker cannot reach a lower point on the mountain range without first going up. Since she does not have a full picture of the mountain range, she will be inclined to stop there. This is precisely the behavior of most iterative algorithms—they will stop at a point when there is no other point with a lower loss function value nearby. However, it is clear that the point with value $\mathcal{L}(\bar{w})$ is not the lowest one on the mountain range. In other words, by starting at the leftmost black dot, most iterative algorithms will stop at the sub-optimal point that corresponds to the value $\mathcal{L}(\bar{w})$. On the other hand, if the algorithm starts at the rightmost black dot, which corresponds to taking w'' as the initial point with loss function value $\mathcal{L}(w'')$, then it will stop at the lowest point on the mountain range. The parameter value at this point is w^*, which corresponds to the prediction rule that best fits the training data.

In view of the above, a natural question is how to find a good initialization for the learning task at hand. Although there are some general rules-of-thumb for choosing the initialization, finding a good one is very much an art and requires substantial human input and experience.[14] Moreover, since the shape of the loss function depends on both the training data and the type of prediction rules used, an initialization that works well for one setting may not work well for another.

From the brief introduction of supervised learning above, it can be seen that the performance of the prediction rule obtained from a supervised learning process hinges upon three human-dependent factors: the quality of the training data (in particular, the informativeness of the labels), the type of prediction rules used to fit the training data (e.g., the choice of a certain neural network architecture), and the algorithm (including its settings such as the initialization and the rule for finding the next point) used to solve the loss minimization problem associated with the learning task. As such, the prediction rules obtained by two different users will generally be different if they specify any of the above factors differently. Putting it in another way, it is generally difficult to reproduce the outcome of a supervised learning process without knowing how each of the above three factors is specified. In addition, the prediction rule obtained is often neither transparent nor interpretable. Indeed, a human cannot easily explain how an iterative algorithm combines the different features of the data to produce the prediction rule, or how the rule makes predictions, or why the rule makes a certain prediction for a given data sample. Such a *black-box* nature of the prediction rule limits our understanding of the learning task at hand and could have various undesirable consequences.[15]

2.2 Unsupervised Learning

Unlike supervised learning, in which the goal is to learn from the labels of the training data samples a rule that can predict the labels of the unseen test data samples as accurately as possible, unsupervised learning is concerned with the scenario where the training data samples do not have any labels and the goal, loosely speaking, is to uncover *hidden structure* in the data. Such a goal is based on the belief that data generated by physical processes are not

[14] To quote Goodfellow, Bengio, and Courville (n 11) , "Modern initialization strategies are simple and heuristic. Designing improved initialization strategies is a difficult task because neural network optimization is not yet well understood." (p. 293).

[15] In recent years, there has been growing interest in *interpretable ML*, which concerns the design of ML systems whose outputs can be explained; see Christoph Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (https://christophm.github.io/interpretable-ml-book/, 2019) for some recent advances in this direction.

random but rather contain information about the processes themselves.[16] For instance, a picture taken by a camera typically contains a foreground and a background, and one can try to identify the backgrounds in image data for further processing. However, in an unsupervised learning task, there is no external guidance on whether the uncovered structure is correct or not, hence the word "unsupervised". This is in contrast to supervised learning, where one can evaluate the accuracy of the prediction rule by comparing the predicted labels and actual labels of the data samples. Thus, one may say that unsupervised learning has a much less well-defined goal. Nevertheless, it is more typical of how humans learn. Indeed, as Geoffrey Hinton, one of the most prominent researchers in artificial intelligence, put it:[17]

> *When we're learning to see, nobody's telling us what the right answers are—we just look. [...] The brain's visual system requires* 10^{14} *[neural] connections. And you only live for* 10^9 *seconds. So it's no use learning one bit per second. You need more like* 10^5 *bits per second. And there's only one place you can get that much information—from the input itself.*

Moreover, since unsupervised learning does not require labeled data—which, as previously mentioned, are not only more expensive to obtain but also quite limited due to the substantial human efforts involved—it is applicable to many more settings. As a first illustration, let us consider one of the most fundamental tasks in unsupervised learning—*clustering*.

2.2.1 Clustering

Roughly speaking, the goal of clustering is to divide the data samples into groups, so that those with similar characteristics belong to the same group and those with different characteristics are separated into different groups. The discovered clusters can then inform the decisions of human users. Clustering is a data analysis technique that has many applications, such as customer segmentation (the process of identifying groups of customers with similar characteristics so that targeted marketing can be carried out)[18] and image segmentation (the process of dividing images into regions so that each region is largely homogeneous).[19]

One may note the similarity of clustering with classification (recall the example shown in Figure 3). However, there is a fundamental difference

[16] DeLiang Wang, 'Unsupervised Learning: Foundations of Neural Computation—A Review' (2001) 22(2) AI Magazine 101.

[17] Pam Frost Groder, 'Neural Networks Show New Promise for Machine Vision' (2006) 8(6) Computing in Science & Engineering 4.

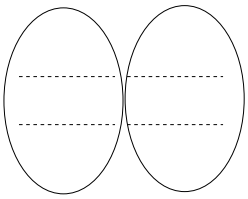
[18] Michael J Shaw and others, 'Knowledge management and data mining for marketing' (2001) 31(1) Decision Support Systems 127.

[19] Richard Szeliski, *Computer Vision: Algorithms and Applications* (Springer-Verlag 2011).

between the two. In a classification task, each training data sample is labeled with its category. As such, we know exactly how many categories are there and what similar data samples look like. By contrast, in a clustering task, it is not clear how many groups one should divide the data samples into and how to define similarity (or dissimilarity) between two samples. Thus, depending on the notion of similarity used, it is entirely possible to come up with different but equally convincing groupings of the same set of data. To demonstrate such a possibility, let us consider dividing the points below (which constitute the data samples) into two groups.[20]

If one prefers not to separate nearby points, then the points should be divided into the two groups shown in Figure 6a. However, if one prefers not to have far-away points belonging to the same group, then the points should be divided into the two groups shown in Figure 6b.

(a) Nearby points should not be separated



(b) Far-away points should not share the same group

Figure 6: Clustering points into two clusters.

The existence of multiple different clustering criteria motivates the development of various clustering algorithms. In general, human input is needed to define a suitable criterion for the clustering task at hand. Once the criterion is fixed, further human input is needed to either choose an existing algorithm (if available) or design a new one to compute a desired clustering of the data samples.

2.2.2 Generative Modeling

As another illustration, let us turn to *generative modeling*—an unsupervised learning task that has attracted much attention in recent years. Besides being used to inform decisions, the information extracted from the training data can also be used to build a model (in the form of an algorithm) for generating new data samples that resemble or are highly related to the training data. The task of building such a model is known as generative

[20]This example is taken from Chapter 22 of Shalev-Shwartz and Ben-David (n 1).

modeling. It lies at the core of many intriguing applications, such as image generation (e.g., to generate highly realistic synthetic photographs that are practically indistinguishable from real ones)[21] and poem generation (e.g., to generate a poem that "describes" the scene of an input image).[22] Currently, one of the most powerful approaches to generative modeling is to use a neural network architecture called *generative adversarial network* (GAN).

A GAN consists of two components, namely the *generator* and the *discriminator*. The generator produces new data samples that are supposed to resemble the training data (the fake data). These generated data samples are then passed along with some training data samples (the real data) to the discriminator, whose task is to classify the data samples it receives as either real or fake. As the word "adversarial" suggests, these two components can be viewed as two players in a game, in which the generator aims to make the real and fake data samples indistinguishable to the discriminator, while the discriminator aims to correctly classify the data samples that are being passed to it. The two components interact in rounds. At the end of each round, the generator is updated depending on how well it fools the discriminator, while the discriminator is updated depending on how well it classifies the real and fake data samples. The interaction ends when the discriminator is no longer able to distinguish between the real and fake data.[23] At this point we say that the GAN is *trained*, and the resulting generator can be used as the generative model. Figure 7 shows the schematic view of a GAN.

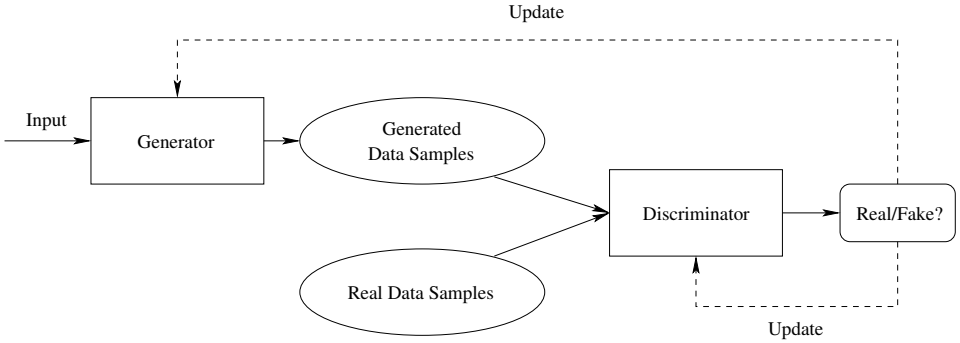


Figure 7: Schematic view of a GAN.

In more detail, both the generator and discriminator are represented by

[21]For a state-of-the-art approach, see Tero Karras and others, 'Analyzing and Improving the Image Quality of StyleGAN' (2019) ⟨https://arxiv.org/abs/1912.04958⟩ Readers who are interested in seeing its performance can visit https://thispersondoesnotexist.com/ (for human face generation) or https://thiscatdoesnotexist.com/ (for cat photo generation).

[22]Bei Liu and others, 'Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training' [2018] Proceedings of the 26th ACM International Conference on Multimedia 783.

[23]One can formalize this by comparing the classification accuracy of the discriminator with that of random guess, which is 1/2.

neural networks. From the discussion in Section 2.1, we know that each of these neural networks has its own set of parameters. To measure how well these two networks perform, two loss functions are introduced, one for the generator (denoted by $\mathcal{L}_G$) and one for the discriminator (denoted by $\mathcal{L}_D$). These loss functions take the parameters of both the generator and discriminator networks as input. However, since the generator and discriminator are competing against each other, each can only control the parameters of its own network. In particular, only after both the generator and discriminator fix the values of the parameters of their own networks would the values of the loss functions $\mathcal{L}_G$ and $\mathcal{L}_D$ be known. Such a setting can best be understood as a game between the generator and discriminator, where each player's move is to choose the values of the parameters it controls, and the payoffs to the players are given by the corresponding values of the loss functions and are known only after both players make their moves. The goal of the game is to find the values of the parameters of the generator and discriminator networks so that the loss functions $\mathcal{L}_G$ and $\mathcal{L}_D$ are minimized. This gives rise to a computational problem that can again be solved by iterative algorithms.

Naturally, the performance of a GAN depends on the architectures of the generator and discriminator networks, the choice of the loss functions $\mathcal{L}_G$ and $\mathcal{L}_D$ and the iterative algorithm used to minimize them, and the training data. While there are some standard choices for the network architectures, loss functions, and the algorithm for minimizing the loss functions,[24] a human user will have to come up with the training data and the initialization strategy for the algorithm. Without knowing either of these two ingredients, it is virtually impossible to reproduce the generator obtained from a trained GAN.

In view of its applications in generative modeling, particularly on content (e.g., photos, artworks, poems, etc.) generation, one may ask whether GANs have the intelligence to do creation on their own. Our discussions above suggest that the answer is no. Indeed, the training of a GAN relies on a number of ingredients supplied by a human user. The computer only executes the instructions of the human user to identify the patterns in and extract information from the training data. The generator obtained from a trained GAN can thus be viewed as a nonlinear function that is created using the ingredients supplied by the human user. However, it is generally difficult to pin down precisely how the function depends on those ingredients, as the process of creating the function is too complex for humans to reason about using the currently available tools.

[24] Ian Goodfellow, 'NIPS 2016 Tutorial: Generative Adversarial Networks' (2017) ⟨https://arxiv.org/abs/1701.00160⟩.

2.3 Reinforcement Learning

Reinforcement learning refers to the scenario in which an *agent*[25] learns by interacting with its environment over time to achieve a certain goal. The interaction involves the agent taking actions to change the *state* of the environment and receiving *feedback* in the form of rewards and penalties from the environment, while the goal is typically to maximize the total cumulative reward. To have a more concrete understanding of the above concepts, let us consider the familiar game of tic-tac-toe.

Illustration: Tic-Tac-Toe. In the classic setting of tic-tac-toe, two players take turns playing on a 3-by-3 board. One player places an "$\times$" in an unoccupied slot of the board when it is her turn, while the other player places an "$\circ$". A player wins when her symbol appears three in a row, either horizontally, vertically, or diagonally.

With the above setup, each player is an agent. A state of the game corresponds to a configuration (i.e., placement of the $\times$'s and $\circ$'s) of the board. The actions available to an agent on her turn are the different ways she can place her symbol in the current configuration of the board. Hence, the game changes to a new configuration after the play of each player. There could be many different ways to define the reward of an action. One possibility is to define it as 1 if the game is won after the action is played, -1 if the game is lost or ends in a draw, and 0 otherwise.

It is worth noting that even in such a simple game, the number of different states is large: Assuming that "$\times$" is played first, there are 5,478 different states of the game! ■

From the above illustration, it can be seen that reinforcement learning is rather different from supervised learning. Indeed, it needs to account for the interactions between the agent and the environment, and it is often impractical to obtain labeled training data that indicate which actions are "correct" in which state of the environment. Reinforcement learning is also different from unsupervised learning, in that it aims to maximize a certain reward function rather than to uncover hidden structure. Generally speaking, in a reinforcement learning task, an agent only knows the reward of the action it has taken but not whether that action has the most reward. Thus, the agent has to try out different actions in order to discover the one with the most reward. In the process, however, the agent could incur penalties. In addition, the action taken by the agent at present time may affect not only the immediate reward but also the actions that are available afterwards and hence the future rewards. These features naturally pose a great challenge to the agent. On one hand, the agent should *exploit* what it has learned from the environment by repeating actions that it has taken

[25]The term "agent" refers to a generic decision-making entity, such as a computer program.

before and found to produce reward. On the other hand, the agent should *explore* the environment by trying actions that it has not taken before in order to discover ones with better reward. Therefore, to be successful in a reinforcement learning task, the agent must carefully manage the trade-off between exploration and exploitation.

There are various algorithms for solving reinforcement learning problems. One family is the *evolutionary methods*.[26] These methods are inspired by biological evolution and proceed in three steps. First, a collection of initial solutions is generated. Here, a solution, which is commonly referred to as a *policy* in the literature, takes the form of a function that specifies the action to be taken by the agent from each state. Next, the solutions undergo "reproduction", meaning that new solutions are generated, e.g., by either combining (known as *recombination*) or modifying (known as *mutation*) existing ones in a certain way. Then, a "natural selection" of the solutions is performed, in which solutions that yield the most reward are carried over to the next generation. These solutions will undergo another round of reproduction and the whole process repeats until certain stopping criterion is met. There are various evolutionary methods in the literature, which differ in their implementations of the three steps above. These methods can be effective when the number of states and number of actions available at each state are small. Nevertheless, they do not make use of the information contained in the state-action relationships of the reinforcement learning problem at hand. For instance, the natural selection step selects solutions that lead to a high cumulative reward. However, for those solutions, it does not reveal which actions taken in which states are crucial to getting the high cumulative reward. Thus, evolutionary methods can be quite inefficient.[27]

Another family of algorithms that can address this shortcoming is the *value function methods*. Roughly speaking, these methods also proceed in three steps, and they differ in their implementations of these steps. First, using some initial policy, a sequence of states, together with the actions taken and the corresponding rewards earned along the way, is generated. Next, the generated information is used to estimate a *value function*, which specifies for each state the total reward that an agent can expect to earn in the future if it starts from that state. For environments with a huge number of states,[28] the value function is typically approximated by a neural network. In this case, estimating the value function means finding values of

[26] Zhi-Hua Zhou, Yang Yu, and Chao Qian, *Evolutionary Learning: Advances in Theories and Algorithms* (Springer Nature Singapore Pte Ltd 2019).

[27] Richard S Sutton and Andrew G Barto, *Reinforcement Learning: An Introduction* (Second Edition, The MIT Press 2018).

[28] For many contemporary applications such as the board game Go, the number of states can easily exceed the number of atoms in the whole universe (which is roughly 10^{80}). Even with today's supercomputer, which can execute roughly 10^{17} calculations per second, it will take more than 10^{63} seconds, or 10^{55} years, to enumerate all the states.

the parameters of the neural network that best fit the information collected in the first step. Lastly, the estimated value function is used to update the policy, and the whole process repeats until certain stopping criterion is met. It should be noted that reward and value are two different notions. The former measures the *immediate* merit of an action, while the latter measures the *long-term* merit of a state by taking into account the possible subsequent states and the rewards of the actions that lead to those states. Thus, value function methods can evaluate the merit of individual states and hence can better exploit the state-action relationships of the problem at hand.[29]

Recently, the use of reinforcement learning techniques has led to some very impressive advances in board game-playing (e.g., AlphaGo),[30] video game-playing (e.g., StarCraft),[31] and autonomous driving,[32] just to name a few. As most applications of interest give rise to reinforcement learning tasks that have an astronomical number of states, careful implementation of algorithms and significant computational resources are essential to getting good results. Both of these factors require substantial human input and cannot be easily reproduced.

3 Closing Remarks

In this chapter, I gave an overview of three types of ML problems and discussed the technical elements in each. A theme that is common in all three types of problems is the use of certain algorithms to extract information from data so as to enable humans to perform complex tasks. Whether the learning process yields useful results depends mainly on the formulation of the learning task at hand (e.g., for supervised learning, the type of prediction rule used for classification; for unsupervised learning, the criterion used to define clusters in the data; for reinforcement learning, the reward and penalty used to train a self-driving car), the algorithm used to tackle the formulation and its implementation details (e.g., initialization strategy), and the volume and quality of the data (which represent past knowledge or experience). As explained in this chapter, the above factors, especially the last two, rely heavily on human users' input. In particular, it is generally difficult to reproduce the outcome of a learning process without knowing how each of the factors is specified. Moreover, the outcome is typically given as a black box, which lacks transparency and interpretability. It is

[29] Sutton and Barto (n 27).

[30] David Silver and others, 'Mastering the game of Go without human knowledge' (2017) 550 Nature 354.

[31] Oriol Vinyals and others, 'Grandmaster level in StarCraft II using multi-agent reinforcement learning' (2019) 575 Nature 350.

[32] Jack Stilgoe, 'Self-driving cars will take a while to get right' (2019) 1 Nature Machine Intelligence 202.

often unclear to humans what features of the training data are used by an ML algorithm to produce the output and how the output accomplishes its objective.

Due to their ability to perform tasks that are beyond human capabilities, modern ML systems are often deemed "intelligent" in the sense that they can create or reason on their own. In reality, the power of these systems is limited by how the learning tasks are formulated and what data are used to train them. As it turns out, such a limitation could have far-reaching consequences, legal, ethical, and otherwise. For instance, since ML algorithms take the training data as input, they will pick up biases in the data and encode them in their output. In a recent study, it has been shown that "standard machine learning can acquire stereotyped biases from textual data that reflect everyday human culture".[33] On one hand, such a finding suggests the possibility of using ML techniques to study and identify prejudicial behavior in humans. On the other hand, it also means that ML-based decision support tools can give discriminatory results. A case in point is Amazon's ML-based recruiting tool, which aims to automate the process of reviewing job applicants' résumés and searching for top talents.[34] The tool took the résumés submitted to the company over a 10-year period as training data. However, most of these résumés were from men. As a result, the tool tends to penalize résumés from female applicants. Although the problem was later identified and attempts to make the tool more gender-neutral were made, the black-box nature of ML processes means that it is difficult to rule out other form of biases in the resulting tool. The need to tackle the issue of bias in ML systems in part drives the recent growth in research on interpretable and fair ML.[35] Another example is adversarial attacks on ML systems—i.e., the use of carefully designed data samples to force the ML systems to commit an error. For instance, it has been found that various image classification systems obtained by training standard neural network architectures using different sets of training data can be fooled by adversarially designed images. In some cases, the adversarial image is just a slight perturbation of a correctly classified image and is essentially the same as the latter from a human perspective.[36] Such a finding shows that the behavior of ML systems

[33]Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan, 'Semantics derived automatically from language corpora contain human-like biases' (2017) 356(6334) Science 183.

[34]Reuters, *Amazon scraps secret AI recruiting tool that showed bias against women* (2018) ⟨https : / / www . reuters . com / article / us - amazon - com - jobs - automation - insight / amazon - scraps - secret - ai - recruiting - tool - that - showed - bias - against - women - idUSKCN1MK08G⟩.

[35]Molnar (n 15); Morgan Gregory, 'What Does Fairness in AI Mean?' (15 January 2020) ⟨https://www.forbes.com/sites/googlecloud/2020/01/15/what-does-fairness-in-ai-mean/⟩.

[36]Readers can refer to Samuel G Finlayson and others, 'Adversarial attacks on medical machine learning' (2019) 363(6433) Science 1287, for examples of adversarial attacks on medical imaging systems.

are not only very different from that of humans but can also be manipulated in ways that have negative consequences. It is now an active research area to develop ML systems that are robust against adversarial attacks.[37]

Although the aforementioned efforts to overcome the limitations of ML systems do lead us closer to being able to explain their inner workings and interpret their outputs, there is still much work to be done and it is entirely possible that the development of new, more complex ML systems can outpace these efforts.

[37]See, e.g., Gean T Pereira and André CPLF de Carvalho, 'Bringing robustness against adversarial attacks' (2019) 1 Nature Machine Intelligence 499.

Bibliography

Article References

Bacham D and Zhao JY, 'Machine Learning: Challenges, Lessons, and Opportunities in Credit Risk Modeling' (2017) 9 Moody's Analytics Risk Perspectives: Managing Disruption 30.

Caliskan A, Bryson JJ, and Narayanan A, 'Semantics derived automatically from language corpora contain human-like biases' (2017) 356(6334) Science 183.

Finlayson SG and others, 'Adversarial attacks on medical machine learning' (2019) 363(6433) Science 1287.

Groder PF, 'Neural Networks Show New Promise for Machine Vision' (2006) 8(6) Computing in Science & Engineering 4.

Litjens G and others, 'A survey on deep learning in medical image analysis' (2017) 42 Medical Image Analysis 60.

Liu B and others, 'Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training' [2018] Proceedings of the 26th ACM International Conference on Multimedia 783.

Pereira GT and de Carvalho ACF, 'Bringing robustness against adversarial attacks' (2019) 1 Nature Machine Intelligence 499.

Shaw MJ and others, 'Knowledge management and data mining for marketing' (2001) 31(1) Decision Support Systems 127.

Silver D and others, 'Mastering the game of Go without human knowledge' (2017) 550 Nature 354.

Stilgoe J, 'Self-driving cars will take a while to get right' (2019) 1 Nature Machine Intelligence 202.

Surden H, 'Machine Learning and Law' (2014) 89 Washington Law Review 87.

Vinyals O and others, 'Grandmaster level in StarCraft II using multi-agent reinforcement learning' (2019) 575 Nature 350.

von Ahn L and others, 'reCAPTCHA: Human-Based Character Recognition via Web Security Measures' (2008) 321(5895) Science 1465.

Wang D, 'Unsupervised Learning: Foundations of Neural Computation—A Review' (2001) 22(2) AI Magazine 101.

Book References

Goodfellow I, Bengio Y, and Courville A, *Deep Learning* (http://www.deeplearningbook.org, MIT Press 2016).

Knuth DE, *The Art of Computer Programming. Volume I: Fundamental Algorithms* (Third edition, Addison Wesley Longman 1997).

Molnar C, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (https://christophm.github.io/interpretable-ml-book/, 2019).

Shalev-Shwartz S and Ben-David S, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press 2014).

Sutton RS and Barto AG, *Reinforcement Learning: An Introduction* (Second Edition, The MIT Press 2018).

Szeliski R, *Computer Vision: Algorithms and Applications* (Springer-Verlag 2011).

Zhou Z.-H, Yu Y, and Qian C, *Evolutionary Learning: Advances in Theories and Algorithms* (Springer Nature Singapore Pte Ltd 2019).

Reports

Reuters, *Amazon scraps secret AI recruiting tool that showed bias against women* (2018) ⟨https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G⟩.

Online References

Goodfellow I, 'NIPS 2016 Tutorial: Generative Adversarial Networks' (2017) ⟨https://arxiv.org/abs/1701.00160⟩.

Gregory M, 'What Does Fairness in AI Mean?' (15 January 2020) ⟨https://www.forbes.com/sites/googlecloud/2020/01/15/what-does-fairness-in-ai-mean/⟩.

Karras T and others, 'Analyzing and Improving the Image Quality of StyleGAN' (2019) ⟨https://arxiv.org/abs/1912.04958⟩.

LeCun Y, Cortes C, and Burges CJC, 'The MNIST database of handwritten digits' (2010) ⟨http://yann.lecun.com/exdb/mnist/⟩.

Ruder S, 'An overview of gradient descent optimization algorithms' (2016) ⟨https://arxiv.org/abs/1609.04747⟩.