In "Lifelong Machine Learning" by Z. Chen and B. Liu, 2018. Morgan & Claypool Publishers.

CHAPTER 1

Introduction

Machine learning (ML) has been instrumental for the advance of both data analysis and artificial intelligence (AI). The recent success of deep learning brought ML to a new height. ML algorithms have been applied in almost all areas of computer science, natural science, engineering, social sciences, and beyond. Practical applications are even more widespread. Without effective ML algorithms, many industries would not have existed or flourished, e.g., Internet commerce and Web search. However, the current ML paradigm is not without its weaknesses. In this chapter, we first discuss the classic ML paradigm and its shortcomings, and then introduce *Lifelong ML* (or simply *Lifelong Learning* (LL)) as an emerging and promising direction to overcome those shortcomings with the ultimate goal of building machines that learn like humans.

1.1 CLASSIC MACHINE LEARNING PARADIGM

The current dominant paradigm for ML is to run an ML algorithm on a given dataset to generate a model. The model is then applied in real-life performance tasks. This is true for both supervised learning and unsupervised learning. We call this paradigm *isolated learning* because it does not consider any other related information or the previously learned knowledge. The fundamental problem with this isolated learning paradigm is that it does not retain and accumulate knowledge learned in the past and use it in future learning. This is in sharp contrast to our human learning. We humans never learn in isolation or from scratch. We always retain the knowledge learned in the past and use it to help future learning and problem solving. Without the ability to accumulate and use the past knowledge, an ML algorithm typically needs a large number of training examples in order to learn effectively. The learning environments are typically static and closed. For supervised learning, labeling of training data is often done manually, which is very labor-intensive and time-consuming. Since the world is too complex with too many possible tasks, it is almost impossible to label a large number of examples for every possible task or application for an ML algorithm to learn. To make matters worse, everything around us also changes constantly, and the labeling thus needs to be done continually, which is a daunting task for humans. Even for unsupervised learning, collecting a large volume of data may not be possible in many cases.

In contrast, we humans learn quite differently. We accumulate and maintain the knowledge learned from previous tasks and use it seamlessly in learning new tasks and solving new problems. That is why whenever we encounter a new situation or problem, we may notice that many aspects of it are not really new because we have seen them in the past in some other con-

texts. When faced with a new problem or a new environment, we can adapt our past knowledge to deal with the new situation and also learn from it. Over time we learn more and more, and become more and more knowledgeable and more and more effective at learning. *Lifelong machine learning* or simply *lifelong learning* (LL) aims to imitate this human learning process and capability. This type of learning is quite natural because things around us are closely related and interconnected. Knowledge learned about some subjects can help us understand and learn some other subjects. For example, we humans do not need 1,000 positive online reviews and 1,000 negative online reviews of movies as an ML algorithm needs in order to build an accurate classifier to classify positive and negative reviews about a movie. In fact, for this task, without a single training example, we can already perform the classification task. How can that be? The reason is simple. It is because we have accumulated so much knowledge in the past about the language expressions that people use to praise or to criticize things, although none of those praises or criticisms may be in the form of online reviews. Interestingly, if we do not have such past knowledge, we humans are probably unable to manually build a good classifier even with 1,000 training positive reviews and 1,000 training negative reviews without spending an enormous amount of time. For example, if you have no knowledge of Arabic and someone gives you 2,000 labeled training reviews in Arabic and asks you to build a classifier manually, most probably you will not be able to do it without using a translator.

To make the case more general, we use natural language processing (NLP) as an example. It is easy to see the importance of LL to NLP for several reasons. First, words and phrases have almost the same meaning in all domains and all tasks. Second, sentences in every domain follow the same syntax or grammar. Third, almost all natural language processing problems are closely related to each other, which means that they are inter-connected and affect each other in some ways. The first two reasons ensure that the knowledge learned can be used across domains and tasks due to the sharing of the same expressions and meanings and the same syntax. That is why we humans do not need to re-learn the language (or to learn a new language) whenever we encounter a new application domain. For example, assume we have never studied psychology, and we want to study it now. We do not need to learn the language used in the psychology text except some new concepts in the psychology domain because everything about the language itself is the same as in any other domain or area. The third reason ensures that LL can be used across different types of tasks. For example, a named entity recognition (NER) system has learned that iPhone is a product or entity, and a data mining system has discovered that every product has a price and the adjective "expensive" describes the price attribute of an entity. Then, from the sentence *"The picture quality of iPhone is great, but it is quite expensive,"* we can safely extract "picture quality" as a feature or attribute of iPhone, and detect that "it" refers to iPhone not the picture quality with the help of those pieces of prior knowledge. Traditionally, these problems are solved separately in isolation, but they are all related and can help each other because the results from one problem can be useful to others. This situation is common for all NLP tasks. Note that we regard anything from unknown to known as a piece of knowledge. Thus, a learned model is a

piece of knowledge and the results gained from applying the model are also knowledge, although they are different kinds of knowledge. For example, iPhone being an entity and picture quality being an attribute of iPhone are two pieces of knowledge.

Realizing and being able to exploit the sharing of words and expressions across domains and inter-connectedness of tasks are still insufficient. A large quantity of knowledge is often needed in order to help the new task learning effectively because the knowledge gained from one previous task may contain only a tiny bit or even no knowledge that is applicable to the new task (unless the two tasks are extremely similar). Thus, it is important to learn from a large number of diverse domains to accumulate a large amount of diverse knowledge. A future task can pick and choose the appropriate past knowledge to use to help its learning. As the world also changes constantly, the learning should thus be continuous or lifelong, which is what we humans do.

Although we used NLP as an example, the general idea is true for any other area because again things in the world are related and inter-connected. There is probably nothing that is completely unrelated to anything else. Thus, knowledge learned in the past in some domains can be applied in some other domains with similar contexts. The classic isolated learning paradigm is unable to perform such LL. As mentioned earlier, it is only suitable for narrow and restricted tasks in closed environments. It is also probably not sufficient for building an intelligent system that can learn continually to achieve close to the human level of intelligence. LL aims to make progress in this very direction. With the popularity of robots, intelligent personal assistants, and chatbots, LL is becoming increasingly important because these systems have to interact with humans and/or other systems, learn constantly in the process, and retain the knowledge learned in their interactions in the ever-changing environments to enable them to learn more and to function better over time.

1.2 MOTIVATING EXAMPLES

In the above, we motivated LL from the perspective of human learning and NLP. In this section, we use some concrete examples, i.e., sentiment analysis, self-driving cars, and chatbots, to further motivate LL. Our original motivation for studying LL actually stemmed from extensive application experiences in sentiment analysis (SA) in a start-up company several years ago. There are two main tasks that an SA system needs to perform: The first task is usually called aspect extraction, which discovers the entities (e.g., iPhone) and entity attributes/features (e.g., battery life) that people talked about in an opinion document such as an online review. These entities and entity attributes are commonly called *aspects* in SA. The second task is to determine whether an opinion about an aspect (entity or entity attribute) is positive, negative, or neutral [Liu, 2012, 2015]. For example, from the sentence "iPhone is really cool, but its battery life sucks," an SA system should discover that the author is positive about iPhone but negative about iPhone's battery life.

There are two main types of application scenarios. The first type is to analyze consumer opinions about one particular product or service (or a small number of products or services), e.g., iPhone or a particular hotel. This kind of application is highly focused and usually not very difficult. The second type is to analyze consumer opinions about a large number of products or services, e.g., opinions about all products sold on Amazon's or Best Buy's websites. Although compared to the first type, the second type is just a quantity change, in fact, it leads to a sea quality change because the techniques used for the first type may no longer be practical for the second type. Let us look at both the supervised and the unsupervised approaches to performing these tasks.

We first analyze the supervised approach. For the first type of application, it is reasonable to spend some time and effort to manually label a large amount of data for aspect extraction and sentiment classification. Note that these are very different tasks and require different kinds of labeling or annotations. With the labeled training data, we can experiment with different ML models, tune their parameters, and design different features in order to build good models for extraction and for classification. This approach is reasonable because we only need to work on opinions about one product or service. In the unsupervised approach, the common method is to use syntactic rules compiled by humans for aspect extraction. For sentiment classification, the common approach is to use a list sentiment words and phrases (e.g., good, bad, beautiful, bad, horrible, and awful) and syntactic analysis to decide sentiments. Although these methods are called unsupervised, they are not completely domain independent. In different domains, extraction rules could be different because people may have somewhat different ways to express opinions. For sentiment classification, a word may be positive in one domain or even in one particular context but negative in another domain. For example, for the word "quiet," the sentence "this car is very quiet" in the car domain is positive, but the sentence "this earphone is very quiet" is negative in the earphone domain. There are also other difficult issues [Liu et al., 2015b]. If we only need to deal with one or two domains (products or services), we can spend time to handcraft rules and identify those domain specific sentiments in order to produce accurate extraction and classification systems.

However, for the second type of application, these two approaches become problematic because they cannot scale up. Amazon.com probably sells hundreds of thousands, if not more, of different products. To label a large amount of data for each kind of product is a daunting task, not to mention new products are launched all the time. It is well known that labeled training data in one domain does not work well for another domain. Although transfer learning can help, it is inaccurate. Worse still, transfer learning usually requires the human user to provide similar source and target domains; otherwise, it can result in negative transfer, and generate poorer results. Although crowdsourcing may be used for labeling, the quality of the labeled data is an issue. More importantly, most products sold on the Web do not have a lot of reviews, which is insufficient for building accurate classifiers or extractors. For the unsupervised approach, the problem is the same. Every type of product is different. For a human to handcraft extraction

rules and identify sentiment words with domain-specific sentiment polarities is also an almost impossible task.

Although the traditional approach is very difficult for the second type of application, it does not mean there is no possible solution. After working on many projects for clients in a start-up company, we realized that there are a significant amount of sharing of knowledge for both aspect extraction and sentiment classification across domains (or different types of products). As we see reviews of more and more products, new things get fewer and fewer. It is easy to notice that sentiment words and expressions (such as good, bad, poor, terrible, and cost an arm and a leg) are shared across domains. There is also a great deal of sharing of aspects (entities and attributes). For example, every product has the attribute of price, most electronic products have batteries, and many of them also have a screen. It is silly not to exploit such sharing to significantly improve SA to make it much more accurate than without using such sharing but only working on the reviews of each product in isolation.

This experience and intuition led us to try to find a systematic way to exploit the knowledge learned in the past. LL is the natural choice as it is a paradigm that learns continually, retains the knowledge learned in the past, and uses the accumulated knowledge to help future learning and problem solving. LL can be applied to both supervised and unsupervised learning approaches to SA. It can enable sentiment analysis to scale up to a very large number of domains. In the supervised approach, we no longer need a large number of labeled training examples. In many domains, no training data is needed at all because they may already be covered by some other/past domains and such similar past domains can be automatically discovered. In the unsupervised approach, it also enables the system to perform more accurate extraction and sentiment classification because of the shared knowledge. It is also possible to automatically discover those domain-specific sentiment polarities of words in a particular domain. We will see some of the techniques in this book.

Interestingly, this application of LL led to two critical problems, i.e., the correctness of knowledge and the applicability of knowledge. Before using a piece of past knowledge for a particular domain, we need to make sure that the piece of past knowledge is actually correct. If it is correct, we must also make sure that it is applicable to the current domain. Without dealing with these two problems, the results in the new domain can get worse. In the later part of the book, we will discuss some methods for solving these problems in both the supervised and unsupervised settings.

For self-driving cars, the situation is similar. There are again two basic approaches to learning to drive: rule-based approach and learning-based approach. In the rule-based approach, it is very hard to write rules to cover all possible scenarios on the road. The learning-based approach has a similar issue because the road environment is highly dynamic and complex. We use the perception system as an example. For the perception system to detect and recognize all kinds of objects on the road in order to predict potential hazards and dangerous situations, it is extremely hard to train a system based on labeled training data. It is highly desirable that the system can

perform continuous learning during driving and in the process identify unseen objects and learn to recognize them, and also learn their behaviors and danger levels to the vehicle by making use of the past knowledge and the feedback from the surround environment. For example, when the car sees a black patch on the road that it has never seen before, it must first recognize that this is an unseen object and then incrementally learn to recognize it in the future, and to assess its danger level to the car. If the other cars have driven over it (environmental feedback), it means that the patch is not dangerous. In fact, on the road, the car can learn a great deal of knowledge from the cars before and after it. This learning process is thus self-supervised (with no external manual labeling of the data) and never ends. As time goes by, the car becomes more and more knowledgeable and smarter and smarter.

Finally, we use the development of chatbots to further motivate LL. In recent years, chatbots have become very popular due to their widespread applications in performing goal-oriented tasks (like assisting customers in buying products, booking flight tickets, etc.) and accompanying humans to get rid of stress via open-ended conversations. Numerous chatbots have been developed or are under development, and many researchers are also actively working on techniques for chatbots. However, there are still some major weaknesses with the current chatbots that limit the scope of their applications. One serious weakness of the current chatbots is that they cannot learn new knowledge during conversations, i.e., their knowledge is fixed beforehand and cannot be expanded or updated during the conversation process. This is very different from our human conversations. We human beings learn a great deal of knowledge in our conversations. We either learn from the utterances of others, or by asking others if we do not understand something. For example, whenever we encounter an unknown concept in a user question or utterance, we try to gather information about it and reason in our brain by accessing the knowledge in our long-term memory before answering the question or responding to the utterance. To gather information, we typically ask questions to the persons whom we are conversing with because acquiring new knowledge through interaction with others is a natural tendency of human beings. The newly acquired information or knowledge not only assists the current reasoning task, but also helps future reasoning. Thus, our knowledge grows over time. As time goes by, we become more and more knowledgeable and better and better at learning and conversing. Naturally, chatbots should have this LL or continuous learning capability. In Chapter 8, we will see an initial attempt to make chatbots learn during conversations.

1.3 A BRIEF HISTORY OF LIFELONG LEARNING

The concept of *lifelong learning* (LL) was proposed around 1995 in Thrun and Mitchell [1995]. Since then it has been pursued in several directions. We give a brief history of the LL research in each of the directions below.

1. *Lifelong Supervised Learning*. Thrun [1996b] first studied lifelong concept learning, where each previous or new task aims to recognize a particular concept or class using binary classification. Several LL techniques were proposed in the contexts of memory-based learning

and neutral networks. The neural network approach was improved in Silver and Mercer [1996, 2002] and Silver et al. [2015]. Ruvolo and Eaton [2013b] proposed an efficient lifelong learning algorithm (ELLA) to improve the multi-task learning (MTL) method in Kumar et al. [2012]. Here the learning tasks are independent of each other. Ruvolo and Eaton [2013a] also considered LL in an active task selection setting. Chen et al. [2015] proposed an LL technique in the context of Naïve Bayesian (NB) classification. A theoretical study of LL was done by Pentina and Lampert [2014] in the PAC-learning framework. Shu et al. [2017b] proposed a method to improve a conditional random fields (CRF) model during model application or testing. It is like *learning on the job*, which other existing models cannot do. Mazumder et al. [2018] worked along a similar line in the context of human-machine conversation to enable chatbots to continually learn new knowledge in the conversation process.

2. *Continual Learning in Deep Neural Networks*. In the past few years, due to the popularity of deep learning, many researchers studied the problem of continually learning a sequence of tasks in the deep learning context [Parisi et al., 2018a]. Note that LL is also called *continual learning* in the deep learning community. The main motivation of continual learning in deep learning is to deal with the problem of *catastrophic forgetting* when learning a series of tasks [McCloskey and Cohen, 1989]. The focus has been on incrementally learning each new task in the same neural network without causing the neural network to forget the models learned for the past tasks. Limited work has been done on how to leverage the previously learned knowledge to help learn the new task better. This is in contrast to the other LL methods, which emphasize leveraging the past knowledge to help new learning.

3. *Open-world Learning*. Traditional supervised learning makes the *closed-world assumption* that the classes of the test instances must have been seen in training [Bendale and Boult, 2015, Fei and Liu, 2016]. This is not suitable for learning in open and dynamic environments because in such an environment, there are always new things showing up. That is, in the model testing or application, some instances from unseen classes may appear. Open-world learning deals with this situation [Bendale and Boult, 2015, Fei et al., 2016, Shu et al., 2017a]. That is, an open-world learner must be able to build models that can detect unseen classes during testing or the model application process, and also learn the new classes incrementally based on the new classes and the old model.

4. *Lifelong Unsupervised Learning*. Papers in this area are mainly about lifelong topic modeling and lifelong information extraction. Chen and Liu [2014a,b] and Wang et al. [2016] proposed several lifelong topic modeling techniques that mine knowledge from topics produced from many previous tasks and use it to help generate better topics in the new task. Liu et al. [2016] also proposed an LL approach based on recommendation for information extraction in the context of opinion mining. Shu et al. [2016] proposed a lifelong relax-

ation labeling method to solve a unsupervised classification problem. These techniques are all based on meta-level mining, i.e., mining the shared knowledge across tasks.

5. *Lifelong Semi-Supervised Learning*. The work in this area is represented by the NELL (*Never-Ending Language Learner*) system [Carlson et al., 2010a, Mitchell et al., 2015], which has been reading the Web continuously for information extraction since January 2010, and it has accumulated millions of entities and relations.

6. *Lifelong Reinforcement Learning*. Thrun and Mitchell [1995] first proposed some LL algorithms for robot learning which tried to capture the invariant knowledge about each individual task. Tanaka and Yamamura [1997] treated each environment as a task for LL. Ring [1998] proposed a continual-learning agent that aims to gradually solve complicated tasks by learning easy tasks first. Wilson et al. [2007] proposed a hierarchical Bayesian lifelong reinforcement learning method in the framework of Markov Decision Process (MDP). Fernández and Veloso [2013] worked on policy reuse in a multi-task setting. A nonlinear feedback policy that generalizes across multiple tasks is proposed in Deisenroth et al. [2014]. Bou Ammar et al. [2014] proposed a policy gradient efficient LL algorithm following the idea in ELLA [Ruvolo and Eaton, 2013b]. This work was further enhanced with cross-domain lifelong reinforcement learning [Bou Ammar et al., 2015a] and with constraints for safe lifelong reinforcement learning [Bou Ammar et al., 2015c].

LL techniques working in other areas also exist. Silver et al. [2013] wrote an excellent survey of the early LL research published at the AAAI 2013 Spring Symposium on LL.

As we can see, although LL has been proposed for more than 20 years, research in the area has not been extensive. There could be many reasons. Some of the reasons may be as follows. First, the ML research community for the past 20 years has focused on statistical and algorithmic approaches. LL typically needs a systems approach that combines multiple components and learning algorithms. Systems approaches to learning were not in favor. This may partially explain that although the LL research has been limited, closely related paradigms of transfer learning and MTL have been researched fairly extensively because they can be done in a more statistical and algorithmic fashion. Second, much of the past ML research and applications focused on supervised learning using structured data, which are not easy for LL because there is little to be shared across tasks or domains. For example, the knowledge learned from a supervised learning system on a loan application is hard to be used in a health or education application because they do not have much in common. Also, most supervised learning algorithms generate no additional knowledge other than the final model or classifier, which is difficult to use as prior knowledge for another classification task even in a similar domain. Third, many effective ML methods such as SVM and deep learning cannot easily use prior knowledge even if such knowledge exists. These classifiers are black boxes and hard to decompose or interpret. They are generally more accurate with more training data. Fourth, related areas such as transfer learning and MTL were popular partly because they typically need only two and just a few similar tasks and datasets and

do not require retention of explicit knowledge. LL, on the other hand, needs significantly more previous tasks and data in order to learn and to accumulate a large amount of explicit knowledge so that the new learning task can pick and choose the suitable knowledge to be used to help the new learning. This is analogous to human learning. If one does not have much knowledge, it is very hard for him/her to learn more knowledge. The more knowledge that one has, the easier it is for him/her to learn even more. For example, it is close to impossible for an elementary school pupil to learn graphical models. Even for an adult, if he has not studied probability theory, it is in-feasible for him to learn graphical models either.

Considering these factors, we believe that one of the more promising areas for LL is NLP due to its extensive sharing of knowledge across domains and tasks and inter-relatedness of NLP tasks as we discussed above. The text data is also abundant. Lifelong supervised, unsupervised, semi-supervised, and reinforcement learning can all be applied to text data.

1.4 DEFINITION OF LIFELONG LEARNING

The early definition of LL is as follows [Thrun, 1996b]. At any point in time, the system has learned to perform N tasks. When faced with the $(N+1)$th task, it uses the knowledge gained from the past N tasks to help learn the $(N+1)$th task. We extend this definition by giving it more details and additional features. First, an explicit *knowledge base* (KB) is added to retain the knowledge learned from previous tasks. Second, the ability to discover new learning tasks during model application is included. Third, the ability to learn while working (or to learn on the job) is incorporated.

Definition 1.1 *Lifelong learning* (LL) is a continuous learning process. At any point in time, the learner has performed a sequence of N learning tasks, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$. These tasks, which are also called the *previous tasks*, have their corresponding datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N$. The tasks can be of different *types* and from different *domains*. When faced with the $(N+1)$th task $\mathcal{T}_{N+1}$ (which is called the *new* or *current task*) with its data $\mathcal{D}_{N+1}$, the learner can leverage the *past knowledge* in the *knowledge base* (KB) to help learn $\mathcal{T}_{N+1}$. The task may be given or discovered by the system itself (see below). The objective of LL is usually to optimize the performance of the new task $\mathcal{T}_{N+1}$, but it can optimize any task by treating the rest of the tasks as the previous tasks. KB maintains the knowledge learned and accumulated from learning the previous tasks. After the completion of learning $\mathcal{T}_{N+1}$, KB is updated with the knowledge (e.g., intermediate as well as the final results) gained from learning $\mathcal{T}_{N+1}$. The updating can involve consistency checking, reasoning, and meta-mining of higher-level knowledge. Ideally, an LL learner should also be able to:

1. learn and function in the open environment, where it not only can apply the learned model or knowledge to solve problems but also *discover new tasks* to be learned, and
2. learn to improve the model performance in the application or testing of the learned model. This is like that after job training, we still *learn on the job* to become better at doing the job.

We can see that this definition is neither formal nor specific because LL is an emerging field and our understanding of it is still limited. For example, the research community still cannot define what knowledge is formally. We believe that it may be better to leave the definition of LL at the conceptual level rather than having it fixed or formalized. Clearly, this does not prevent us from giving a formal definition when we solve a specific LL problem. Below, we give some additional remarks.

1. The definition indicates five key characteristics of LL:

   (a) continuous learning process,

   (b) knowledge accumulation and maintenance in the KB,

   (c) the ability to use the accumulated past knowledge to help future learning,

   (d) the ability to discover new tasks, and

   (e) the ability to learn while working or to learn on the job.

   These characteristics make LL different from related learning paradigms such as transfer learning [Jiang, 2008, Pan and Yang, 2010, Taylor and Stone, 2009] and multi-task learning (MTL) [Caruana, 1997, Chen et al., 2009, Lazaric and Ghavamzadeh, 2010], which do not have one or more of these characteristics. We will discuss these related paradigms and their differences from LL in detail in Chapter 2.

   Without these capabilities, an ML system will not be able to learn in a dynamic open environment by itself, and will never be truly intelligent. By *open environment*, we mean that the application environment may contain novel objects and scenarios that have not been learned before. For example, we want to build a greeting robot for a hotel. At any point in time, the robot has learned to recognize all existing hotel guests. When it sees an existing guest, it can call him/her by his/her first name and chat. It must also detect any new guests that it has not seen before. On seeing a new guest, it can say hello, ask for his/her name, take many pictures, and learn to recognize the guest. Next time when it sees the new guest again, it can call him/her by his/her first name and chat like an old friend. The real-world road environment for self-driving cars is another very typical dynamic and open environment.

2. Since knowledge is accumulated and used in LL, this definition forces us to think about the issue of prior knowledge and the role it plays in learning. LL thus brings in many other aspects of Artificial Intelligence to ML, e.g., knowledge representation, acquisition, reasoning, and maintenance. Knowledge, in fact, plays a central rule. It not only can help improve future learning, but can also help collect and label training data (self-supervision) and discover new tasks to be learned in order to achieve autonomy in learning. The integration of both data-driven learning and knowledge-driven learning is probably what human learning is all about. The current ML focuses almost entirely on data-driven optimization

learning, which we humans are not good at. Instead, we are very good at learning based on our prior knowledge. It is well-known that for human beings, the more we know the more we can learn and the easier we can learn. If we do not know anything, it is very hard to learn anything. ML research should thus pay more attention to knowledge and build machines that learn like humans.[1]

3. We distinguish two types of tasks.

   (a) *Independent tasks*: Each task $\mathcal{T}_i$ is independent of the other tasks. This means that each task can be learned independently, although due to their similarities and sharing of some latent structures or knowledge, learning $\mathcal{T}_i$ can leverage the knowledge gained from learning previous tasks.

   (b) *Dependent tasks*: Each task $\mathcal{T}_i$ has some dependence on some other tasks. For example, in open-world learning (Chapter 5)[Fei et al., 2016], each new supervised learning task adds a new class to the previous classification problem, and needs to build a new multi-class classifier that is able to classify data from all previous and the current classes.

4. The tasks do not have to be from the same domain. Note that there is still no unified definition of a *domain* in the literature that is applicable to all areas. In most cases, the term is used informally to mean a setting with a fixed feature space where there can be multiple different tasks of the same type or of different types (e.g., information extraction, coreference resolution, and entity linking). Some researchers even use domain and task interchangeably because there is only one task from each domain in their study. We also use them interchangeably in many cases in this book due to the same reason but will distinguish them when needed.

5. The shift to the new task can happen abruptly or gradually, and the tasks and their data do not have to be provided by some external systems or human users. Ideally, a lifelong learner should also be able to find its own learning tasks and training data in its interaction with humans and the environment or using its previously learned knowledge to perform open-world and self-supervised learning.

6. The definition indicates that LL may require a *systems approach* that combines multiple learning algorithms and different knowledge representation schemes. It is unlikely that a single learning algorithm is able to achieve the goal of LL. LL, in fact, represents a large and rich problem space. Much research is needed to design algorithms to achieve each capability or characteristic.

[1] If a learning system can do both data-driven optimization and human-level knowledge-based learning, we may say that it has achieved some kind of *super learning* capability, which may also mean that it has reached some level of *artificial general intelligence* (AGI) because we humans certainly cannot do learning based on large scale data-driven optimization or remember a large quantity of knowledge in our brain as a machine can.

Based on Definition 1.1, we can outline a general process of LL and an LL system architecture, which is very different from that of the current *isolated learning* paradigm with only a single task T and dataset D. Figure 1.1 illustrates the classic isolated learning paradigm, where the learned model is used in its intended application.

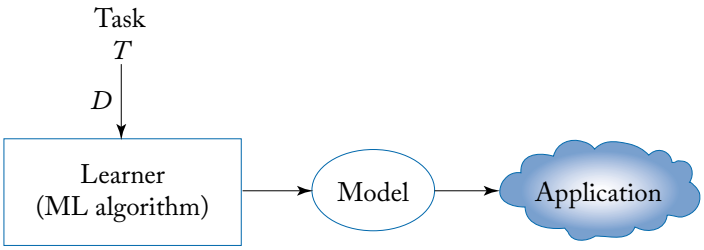

Figure 1.1: The classic machine learning paradigm.

The new LL system architecture is given in Figure 1.2. Below, we first describe the key components of the system and then the LL process. We note that this general architecture is for illustration purposes. Not all existing systems use all the components or sub-components. In fact, most current systems are much simpler. Moreover, there is still not a generic LL system that can perform LL in all possible domains for all possible types of tasks. In fact, we are still far from that. Unlike many ML algorithms such as SVM and deep learning, which can be applied to any learning task as long as the data is represented in a specific format required by these algorithms, current LL algorithms are still specific to some types of tasks and data.

1. **Knowledge Base (KB)**: It is mainly for storing the previously learned knowledge. It has a few sub-components.

   (a) *Past Information Store* (PIS): It stores the information resulted from the past learning, including the resulting models, patterns, or other forms of outcome. PIS may involve sub-stores for information such as (1) the original data used in each previous task, (2) intermediate results from each previous task, and (3) the final model or patterns learned from each previous task. As for what information or knowledge should be retained, it depends on the learning task and the learning algorithm. For a particular system, the user needs to decide what to retain in order to help future learning.

   (b) *Meta-Knowledge Miner* (MKM). It performs meta-mining of the knowledge in the PIS and in the meta-knowledge store (see below). We call this *meta-mining* because it mines higher-level knowledge from the saved knowledge. The resulting knowledge is stored in the Meta-Knowledge Store. Here multiple mining algorithms may be used to produce different types of results.

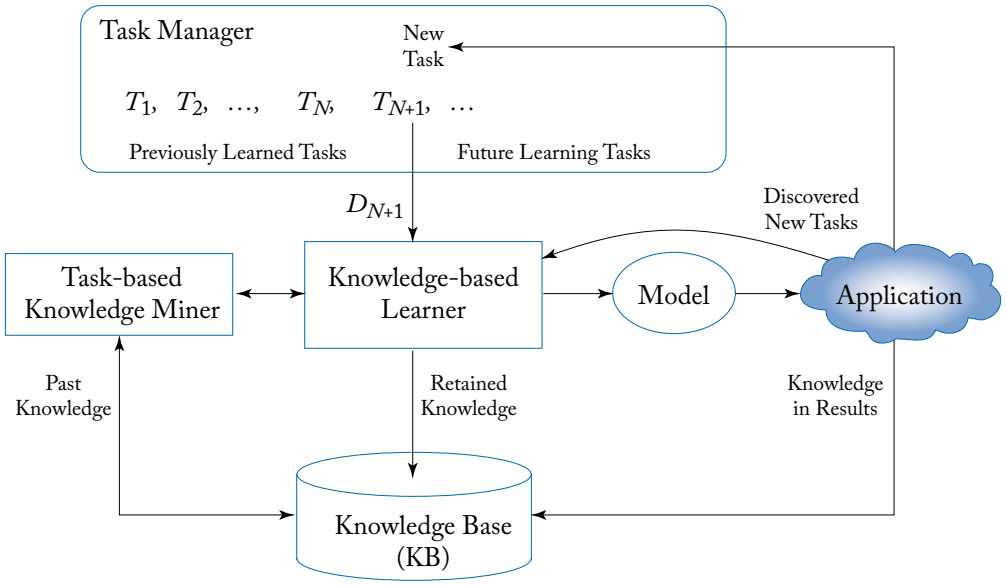

Figure 1.2: The lifelong machine learning system architecture.

(c) *Meta-Knowledge Store* (MKS): It stores the knowledge mined or consolidated from PIS and also from MKS itself. Some suitable knowledge representation schemes are needed for each application.

(d) *Knowledge Reasoner* (KR): It makes inference based on the knowledge in MKB and PIS to generate more knowledge. Most current systems do not have this sub-component. However, with the advance of LL, this component will become increasingly important.

Since the current LL research is still in its infancy, as indicated above, none of the existing systems has all these sub-components.

2. **Knowledge-Based Learner (KBL)**: For LL, it is necessary for the learner to be able to use prior knowledge in learning. We call such a learner a *knowledge-based learner*, which can leverage the knowledge in the KB to learn the new task. This component may have two sub-components: (1) *Task knowledge miner* (TKM), which makes use of the raw knowledge or information in the KB to mine or identify knowledge that is appropriate for the current task. This is needed because in some cases, KBL cannot use the raw knowledge in the KB directly but needs some task-specific and more general knowledge mined from the KB [Chen and Liu, 2014a,b], and (2) the *learner* that can make use of the mined knowledge in learning.

3. **Task-based Knowledge Miner (TKM)**: This module mines knowledge from the KB specifically for the new task.

4. **Model**: This is the learned model, which can be a prediction model or classifier in supervised learning, clusters or topics in unsupervised learning, a policy in reinforcement learning, etc.

5. **Application**: This is the real-world application for the model. It is important to note that during model application, the system can still learn new knowledge (i.e., "knowledge in results"), and possibly discover new tasks to be learned. Application can also give feedback to the knowledge-based learner for model improvement.

6. **Task Manager (TM)**: It receives and manages the tasks that arrive in the system, handles the task shift, and presents the new learning task to the KBL in a lifelong manner.

Lifelong Learning Process: A typical LL process starts with the Task Manager assigning a new task to the KBL (the task can be given or discovered automatically). KBL then works with the help of the past knowledge stored in the KB to produce an output model for the user and also send the information or knowledge that needs to be retained for future use to the KB. In the application process, the system may also discover new tasks and learn while working (learn on the job). Some knowledge gained in applications can also be retained to help future learning.

1.5 TYPES OF KNOWLEDGE AND KEY CHALLENGES

Definition 1.1 does not give any detail about what knowledge or its representation form is in the KB. This is mainly due to our limited understanding. There is still no well-accepted definition of knowledge or its general representation scheme. In the current LL research, past knowledge usually serves as some kind of prior information (e.g., prior model parameters or prior probabilities) for the new task. Each existing paper uses one or two specific forms of knowledge that are suitable for its proposed techniques and intended applications. For example, some methods use a set of shared latent parameters [Ruvolo and Eaton, 2013b, Wilson et al., 2007] as knowledge. Some directly use model parameters of previous tasks as knowledge [Chen et al., 2015, Shu et al., 2016]. Some use previous model application results as knowledge, e.g., topics from topic modeling [Chen and Liu, 2014a, Chen et al., 2015] and items extracted from previous information extraction models [Liu et al., 2016, Shu et al., 2017b]. Some even use past relevant data as knowledge to augment the new task data [Xu et al., 2018]. Knowledge is usually represented based on how it is used in individual algorithms. There are still no general knowledge representation schemes that can be used across different algorithms or different types of tasks. Definition 1.1 also does not specify how to maintain or update the KB. For a particular LL algorithm and a particular form of shared knowledge, one needs to design a KB and its maintenance or updating methods based on the algorithm and its knowledge representation need.

There are mainly two types of shared knowledge that are used in learning the new task.

1. *Global knowledge*: Many existing LL methods assume that there is a *global latent structure* among tasks that is shared by all tasks [Bou Ammar et al., 2014, Ruvolo and Eaton, 2013b, Thrun, 1996b, Wilson et al., 2007] (Sections 3.2, 3.4, 9.1, 9.2, and 9.3). This global structure can be learned and leveraged in the new task learning. The approaches based on global knowledge transfer and sharing mainly grew out of or inspired by MTL, which jointly optimizes the learning of multiple similar tasks. Such knowledge is more suitable for similar tasks in the same domain because such tasks are often highly correlated or have very similar distributions.

2. *Local knowledge*: Many other methods do not assume such a global latent structure among tasks [Chen and Liu, 2014a,b, Chen et al., 2015, Fei et al., 2016, Liu et al., 2016, Shu et al., 2016, Tanaka and Yamamura, 1997] (Sections 3.5, 5.2, 6.2, 6.3, 7.1, 7.2, 7.3, and 7.4). Instead, during the learning of a new task they pick and choose the pieces of knowledge learned from previous tasks to use based on the need of the current task. This means that different tasks may use different pieces of knowledge learned from different previous tasks. We call such pieces of knowledge the *local knowledge* because they are local to their individual previous tasks and are not assumed to form a coherent global structure. Local knowledge is likely to be more suitable for related tasks from different domains because the shared knowledge from any two domains may be small. But the prior knowledge that can be leveraged by the new task can still be large because the prior knowledge can be from many past domains.

LL methods based on local knowledge usually focus on optimizing the current task performance with the help of past knowledge. They can also be used to improve the performance of any previous task by treating that task as the new/current task. The main advantage of these methods is their flexibility as they can choose whatever pieces of past knowledge that are useful to the new task. If nothing is useful, the past knowledge will not be used. The main advantage of LL methods based on global knowledge is that they often approximate optimality on all tasks, including the previous and the current ones. This property is inherited from MTL. However, when the tasks are highly diverse and/or numerous, this can be difficult.

As the previous learned knowledge is involved, apart from the classic issues about knowledge discussed above (e.g., what knowledge to retain, how to represent and use the knowledge, and how to maintain the KB), there are two other fundamental challenges about knowledge in LL. We will describe how some existing techniques deal with these challenges later in the book.

1. *Correctness of knowledge*: Clearly, using incorrect past knowledge is detrimental to the new task learning. In a nutshell, LL can be regarded as a continuous bootstrapping process. Errors can propagate from previous tasks to subsequent tasks to generate more and more errors. We humans seem to have a good idea of what is correct or what is incorrect. But there is still no satisfactory technique for detecting wrong knowledge. Many existing papers do not deal with this challenge [Silver and Mercer, 2002, Silver et al., 2015, Thrun,

1996b] or deal with it implicitly to some extent [Ruvolo and Eaton, 2013b, Wilson et al., 2007]. There are also many papers that deal with the challenge explicitly [Chen and Liu, 2014a,b, Chen et al., 2015, Liu et al., 2016, Mitchell et al., 2015, Shu et al., 2016]. For example, one strategy is to find those pieces of knowledge that have been discovered in many previous tasks/domains [Chen and Liu, 2014a,b, Chen et al., 2015, Shu et al., 2016]. Another strategy is to make sure that the piece of knowledge is discovered from different contexts using different techniques [Mitchell et al., 2015]. Although these and other strategies are useful, they are still not satisfactory because of two main issues. First, they are not foolproof because they can still produce wrong knowledge. Second, they have low recall because most pieces of correct knowledge cannot pass these strategies and thus cannot be used subsequently, which prevents LL from producing even better results. We will detail these strategies when we discuss the related papers.

2. *Applicability of knowledge*. Although a piece of knowledge may be correct in the context of some previous tasks, it may not be applicable to the current task. Application of inappropriate knowledge has the same negative consequence as the above case. Again, we humans are quite good at recognizing the right context for the application of a piece of knowledge, which is very difficult for automated systems. Again, many papers do not deal with the challenge, however, some do, e.g., Chen and Liu [2014a], Chen et al. [2015], Shu et al. [2016], and Xu et al. [2018]. We will describe them when we discuss these papers as they are quite involved.

Clearly, the two challenges are closely related. It is seemingly that we only need to be concerned with the applicability challenge regardless whether the knowledge is correct or not because if a piece of knowledge is not correct, it cannot be applicable to the new task. This is often not the case because in deciding applicability, we may just be able to decide whether a new task or domain context is similar to some older tasks or domain contexts. If so, we can use the knowledge gained from those older tasks. Then we must make sure that the knowledge from those older tasks is correct.

1.6 EVALUATION METHODOLOGY AND ROLE OF BIG DATA

Unlike the classic isolated learning where the evaluation of a learning algorithm is based on training and testing using data from the same task/domain, LL needs a different evaluation methodology because it involves a sequence of tasks and we want to see improvements in the learning of new tasks. Experimental evaluation of an LL algorithm in the current research is commonly done using the following steps.

1. *Run on the data from the previous tasks*: We first run the algorithm on the data from a set of previous tasks, one at a time in a given sequence, and retain the knowledge gained

in the KB. Obviously, there can be multiple variations or versions of the algorithm (e.g., with different types of knowledge used and more or less knowledge used) that can be experimented with.

2. *Run on the data of the new task*: We then run the LL algorithm on the new task data by leveraging the knowledge in the KB.

3. *Run baseline algorithms*: For comparison, we run some baseline algorithms. There are usually two kinds of baselines. The first kind are algorithms that perform isolated learning on the new data without using any past knowledge. The second kind are existing LL algorithms.

4. *Analyze the results*: This step compares the results from steps 2 and 3 and analyzes the results to make some observations, e.g., to show that the results from the LL algorithm in step 2 are superior to those from the baselines in step 3.

There are several additional considerations in carrying out an LL experimental evaluation.

1. *A large number of tasks*: A large number of tasks and datasets are needed to evaluate an LL algorithm. This is because the knowledge gained from a few tasks may not be able to improve the learning of the new task much as each task may only provide a very small amount of knowledge that is useful to the new task (unless all the tasks are very similar) and the data in the new task is often quite small.

2. *Task sequence*: The sequence of the tasks to be learned can be significant, meaning that different task sequences can generate different results. This is so because LL algorithms typically do not guarantee optimal solutions for all previous tasks. To take the sequence effect into consideration in the experiment, one can try several random sequences of tasks and generate results for the sequences. The results can then be aggregated for comparison purposes. Existing papers mainly use only one random sequence in their experiments.

3. *Progressive experiments*: Since more previous tasks generate more knowledge, and more knowledge in turn enables an LL algorithm to produce better results for the new task, it is thus desirable to show how the algorithm performs on the new task as the number of previous tasks increases.

Note that it is not our intention to cover all possible kinds of evaluations in the current research on LL. Our purpose is simply to introduce the common evaluation methodologies. In evaluating a specific algorithm, one has to consider the special characteristics of the algorithm (such as its assumptions and parameter settings) and the related research in order to design a comprehensive set of experiments.

Role of Big Data in LL Evaluation: It is common knowledge that the more you know the more you can learn and the easier you can learn. If we do not know anything, it is very hard

to learn anything. These are intuitive as each one of us must have experienced this in our lives. The same is true for a computer algorithm. Thus, it is important for an LL system to learn from a diverse range and a large number of domains to give the system a wide vocabulary and a wide range of knowledge so that it can help learn in diverse future domains. Furthermore, unlike transfer learning, LL needs to automatically identify the pieces of past knowledge that it can use, and not every past task/domain is useful to the current task. LL experiments and evaluation thus require data from a large number of domains or tasks and consequently large volumes of data. Fortunately, big datasets are now readily available in many applications such as image and text that can be used in LL evaluations.

1.7 OUTLINE OF THE BOOK

This book introduces and surveys this important and emerging field. Although the body of literature is not particularly large, related papers are published in a large number of conferences and journals. There is also a large number of papers that do not exhibit all the characteristics of LL, but are related to it to some extent. It is thus hard, if not impossible, to cover all of the important work in the field. As a result, this book should not be taken to be an exhaustive account of everything in the field.

The book is organized as follows. In Chapter 2, we discuss some related ML paradigms to set the stage and background. We will see that these existing paradigms are different from LL because they lack one or more of the key characteristics of LL. However, all these paradigms involve some forms of knowledge sharing or transfer across tasks and can even be made continual in some cases. Thus, we regard LL as an advanced ML paradigm that extends these existing paradigms in the progression of making ML more intelligent and closer to human learning.

In Chapter 3, we focus on discussing existing research on supervised LL, where we will see a fairly detailed account of some early and more recent supervised LL methods. In Chapter 4, we continue the discussion of supervised LL but in the context of deep neural networks (DNNs), where LL is sometimes also called *continual learning*. The main goal in this context is to solve the *catastrophic forgetting* problem in deep learning when learning multiple tasks. Chapter 5 is another chapter related to supervised learning. However, as its name suggests, this type of learning learns in the open-world, where the test data may contain instances from unseen classes (not seen in training). This is in contrast to the classic closed-world learning, where all test classes have appeared in training.

In Chapter 6, we discuss lifelong topic models. In these models, discovered topics from previous tasks are mined to extract reliable knowledge that can be exploited in the new model inferencing to generate better topics for the new task. Chapter 7 discusses lifelong information extraction. Information extraction is a very suitable problem for LL because information extracted in the past is often quite useful for future extraction due to knowledge sharing across tasks and/or domains. Chapter 8 switches topic and discusses a preliminary work about lifelong interactive knowledge learning in human-machine conversation. This is a new direction as

existing chatbots cannot learn new knowledge after they are built or deployed. Lifelong reinforcement learning is covered in Chapter 9. Chapter 10 concludes the book and discusses some major challenges and future directions of the LL research.