CHAPTER

19

Deep Learning

PREAMBLE

The term "deep learning" appears to presume that other kinds of machine-learning activities are "shallow." This is not the case. The previous chapters have exposed you to some very sophisticated methods in predictive analytics (e.g., lag variables for time-series analysis and ensembles of models). The deepness of deep learning (DL) methods refers to the depth of a signal that is modeled by a series of "hidden" layers in a neural net. Current DL technology is restricted to neural net development, but future methods will be elaborations of other algorithms. For now, we can take a "deep dive" into the current technology of DL.

What Is DL?

Like "Big Data," DL has become a buzzword that means many different things to many people. One of the common foci of interest in subjects associated with DL is the "deep" complexity of the human brain. Fig. 19.1 shows a factually incorrect, but logically fascinating expression of very complex cause-and-effect relationships resident in the human brain that functions in the perception of some people as an intricate clock mechanism. The "depth" of the geared relationships in a given cause-and-effect pathway in the brain can include many "gears" operating in tandem to generate thoughts and responses, which result in actions of the body.

Goodfellow et al. (2016) list four key trends in the development of DL:

1. DL has had a long and rich history, but has gone by many names reflecting different philosophical viewpoints, and has waxed and waned in popularity;
2. DL has become more useful as the amount of available training data has increased;
3. DL models have grown in size over time as computer hardware and software infrastructure for DL has improved;
4. DL has solved increasingly complicated applications with increasing accuracy over time.

Handbook of Statistical Analysis and Data Mining Applications
https://doi.org/10.1016/B978-0-12-416632-5.00019-0

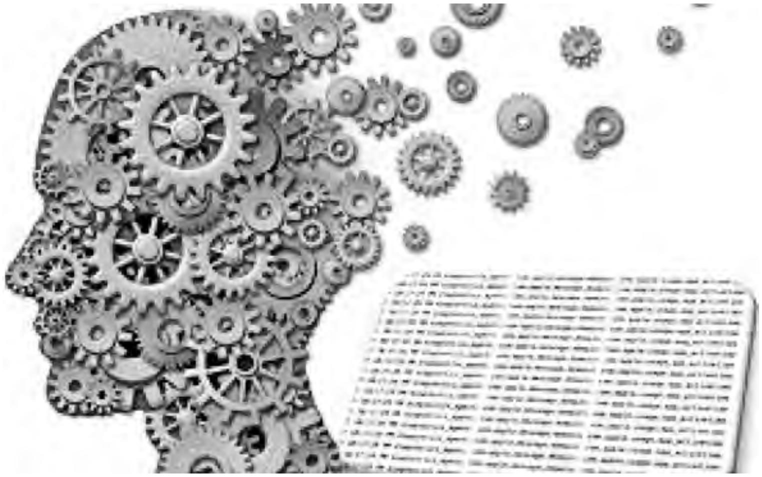

FIG. 19.1 Deep learning "gears" of the human brain. *From https://www.google.com/search?q=deep+learning+images&biw=1920&bih=942&tbm=isch&imgil=SySt4Dda1-wHcM%253A%253BLGqLwznf8oZt8M%253Bhttp%25253A%25252F%25252Fhyperverge.co%25252Fa-beginners-introduction-to-deep-learning%25252F&source=iu&pf=m&fir=SySt4Dda1-wHcM%253A%252CLGqLwznf8oZt8M%252C_&usg=__Nb8nuBtLXzyo7vpMMioD4WEJS3c%3D&ved=0ahUKEwiHwrmV9NPPAhUP-mMKHUCPBsUQyjcINQ&ei=Dnb9V4eDPY_0jwPAnpqoDA#imgrc=EGjVmOLEZOdC0M%3A*.

THE GUIDING CONCEPT OF DL TECHNOLOGY—HUMAN COGNITION

One of the best perspectives from which to understand DL is to view it within the science of human cognition, which studies the intricate "gearing" of the human brain (Fig. 19.1) that facilitates thought. These "gears" are represented in DL algorithms as middle (or "hidden") layers in a neural network architecture. Chapter 1 presents the background and history of data mining (aka predictive analytics), ending with the focus on the most powerful nonlinear analytical system in the universe—the human brain. Data processing machines have been around for a long time, possibly as far back as Ancient Greece (the Antikythera mechanism—Price, 1974). One of the questions that arise inevitably is can a machine think like a human? The quest to explore this question led to the development of artificial intelligence (AI) technology, a branch of cognitive science. Initially, AI practitioners focused on solving problems that are hard for humans, but easy for computers, such as building and using expert systems. "The bigger challenge, however, was to develop techniques for solving problems that are easy for humans, but hard for computers," like image and speech recognition (Schmidhuber, 2015).

The first step in building such AI systems was to mimic rather crudely the way the human brain analyzes sensory input data through the processing of sensory input in a neural network. Fig. 19.2 shows an artist's rendition of a human neural network, composed of a number of neurons (the blob-like forms) connected together with slender pathways (dendrites).

Practitioners in cognitive science recognized that the human brain "thinks" through a series of linked neurons (nerve cells). The dendrites shown in Fig. 19.2 function to pass

FIG. 19.2 Artist's conception of a network of neurons (a neural network) in the human brain. *From https://www.google.com/search?q=deep+learning+images&biw=1920&bih=942&tbm=isch&imgil=SySt4Dda1-wHcM%253A%253BLGqLwznf8oZt8M%253Bhttp%25253A%25252F%25252Fhyperverge.co%25252Fa-beginners-introduction-to-deep-learning%25252F&source=iu&pf=m&fir=SySt4Dda1-wHcM%253A%252CLGqLwznf8oZt8M%252C_&usg=__Nb8nuBtLXzyo7vpMMioD4WEJS3c%3D&ved=0ahUKEwiHwrmV9NPPAhUP-mMKHUCPBsUQyjcINQ&ei=Dnb9V4eDPY_0jwPAnpqoDA#imgrc=bSOmXaxpV08xWM%3A.*

information in the form of electrical impulses between neurons. These electric impulses (or "signals") can become either reduced in strength (attenuated), through biological counterparts to electrical resistors, or increased in strength by biological counterparts of electric capacitors. This signal modulation along the path of the neural impulse flow is analogous to the sequential processing of data in computers. This insight moved AI scientists to develop computer-expressed counterparts to these human neural networks called artificial neural networks, or ANNs.

EARLY ARTIFICIAL NEURAL NETWORKS (ANNs)

The first attempts in this simulation resulted in the development of ANNs focused on distinguishing patterns in single data sets. The biggest difference, however, between these early AI methods and human cognitive abilities is the way humans use experience to adapt to and refine initial sensory patterns.

Early ANNs had only two layers, a layer of data inputs (analogous to human sensory inputs) and a layer of outputs, analogous to the initiation of muscular action. In an ANN, this output consisted of either a numerical prediction (one output node) or a classification list (possibly, with multiple output nodes). The processing included two basic functions: (1) an aggregation function and (2) a response function, mimicking the way the human brain stores sensory input, until a critical threshold is reached, and then "fires" a neural impulse to the next neuron connected in the system (the "network").

Fig. 19.3 shows the general form of a two-layer neural net, composed of an input layer of four neurons (I-1 through I-4), an aggregation function, a response function, and an output neuron (O).

The problem with the early ANNs was that this neural architecture and data processing approach did not function very well when the output response was significantly nonlinear in respect to the inputs. To solve this problem, AI algorithm designers added another

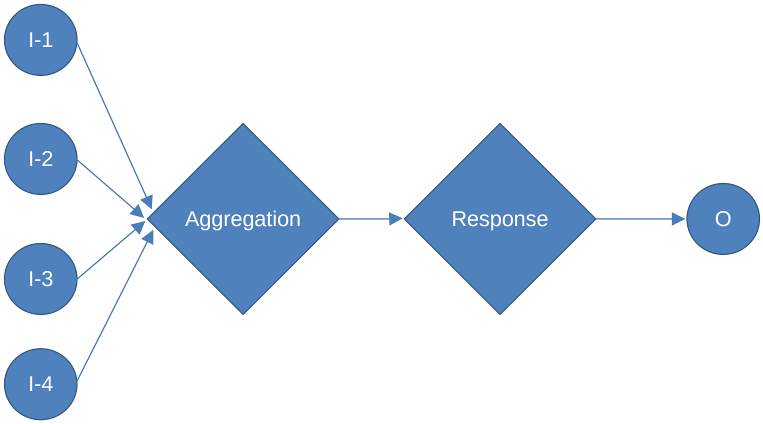

FIG. 19.3 Diagram of a two-layer neural network.

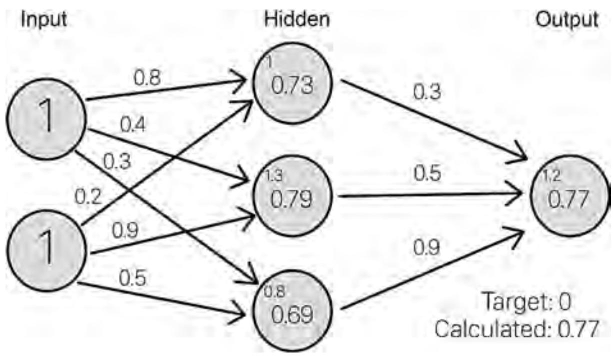

FIG. 19.4 The architecture of the multilayer perceptron (MLP), with node weights.

layer between the input layer and the output layer. They named this structure the "hidden layer." Separate weights between nodes in each layer were optimized. The relationships between the hidden layer and the input and output layers that are resident in these optimized weights functioned to express a large degree of nonlinearity between inputs and output. Fig. 19.4 shows this architecture was termed the multilayer perceptron, or MLP (Rosenblatt, 1961, 1958).

The large number 1 in the two left circles of Fig. 19.4 represents inputs; the numbers on the arrows represent weights; the small numbers in the center circles are the sums of the weight × input values directed to them; the large numbers in the center circles represent the resulting output of the nonlinear activation function controlling the "firing" of that node. The final value of 0.77 in the output node is the sum of the "hidden" (middle) layer node values times the weight associated with its link to the output node, processed through a nonlinear activation function.

This ANN architecture is called a "feed-forward" design, because the data processing steps all flow in one direction from inputs to the output.

HOW ANNs WORK

The general processing of an ANN includes the following:

1. Random assignment of weights for each data input stream of one data subset (called the "training set")
2. Accumulation (or aggregation) of weighted inputs provided by each row in the data set, until a specified threshold is reached, and then "firing" of the accumulated impulse to the output node
3. Initiation of an output prediction or classification, inferred from the relationships between input weights of the variables and the output, following a specified response function (linear, logistic, Poisson, etc.)
4. Calculation of total error in prediction or classification, using another data subset (called the "testing" or "validation" subset)
5. Slight adjustment of the input weights, based on the magnitude and direction of the error
   a. This form of "learning" the most appropriate weights is called "back propagation," because the error is used to modify the inputs that generated it.
   b. Weights of all variables are adjusted for the given iteration through the data set.
6. Processing of the data set again in another iteration through the data set, and a new overall prediction error is calculated
7. This process can span many (often hundreds) of data iterations, each one associated with a slight adjustment in the variable weights, until a "stopping" point has been reached in terms of the following:
   a. Time
   b. Number of data iterations
   c. Minimum overall error is reached

The back-propagation technique is explained further in Chapter 7 (basic algorithms). Other training parameters can be optimized also in a similar manner.

1. The learning rate parameter controls the speed over which the decision landscape is "traversed" in the course of data processing. This parameter is analogous to the speed of a ball rolling over an uneven error surface, symbolized by Fig. 19.5.

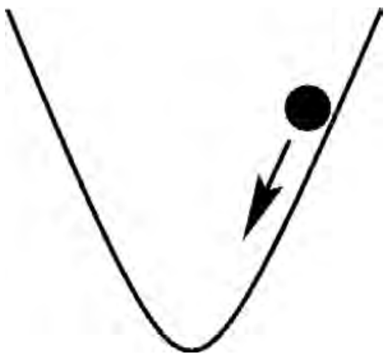

FIG. 19.5 The decision ball is flowing downhill along the error surface.

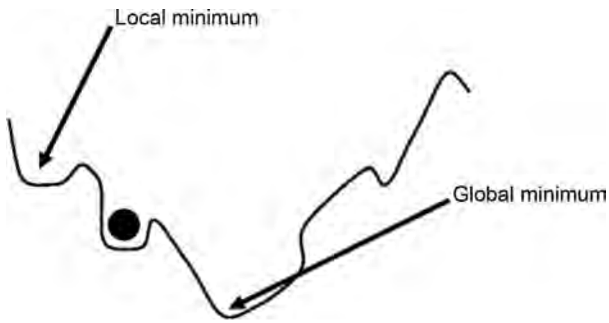

FIG. 19.6 The decision "trapped" in a local minimum on an error surface.

2. The "momentum" parameter (analogous to strength of the tendency for solution optimization to "keep going" over an error gradient, like a ball rolling up or down a hill). The optimization process "seeks" a minimum error point, but the one "found" by the processing might be only a "local" minimum, not a "global" minimum. Some error surfaces are rather uneven and "craggy," which might cause the decision to become "stuck" in a local minimum as in Fig. 19.6.

The momentum parameter helps to "drive" the decision slightly uphill (toward a higher error point) until it "finds" a downward gradient again and continues to the state of global minimum error. The back-propagation process is one of core elements of both original MLP design, and of DL technology.

Problems in the Use of Machine-Learning (ML) Technology

One of the problems with the use of ML technology like ANNs is the tendency for the solutions to get trapped in a local minimum in the error minimization process. Optimizing the momentum of the neural net is one way to reduce that risk. But, this process works only within the training-testing process. Another problem is that error minimization process proceeds to train a relatively accurate model, but it fails miserably when new data are input to it, because the neural net is trained so closely to the training data set, that if cannot distinguish similar patterns in a new data set. This problem is called *over-training.* Patterns in the new data may be sufficiently different than the patterns in the data used to train the model that the prediction accuracy is compromised significantly. This form of error is called "generalization error." Several processing strategies can reduce the generalization error. This type of process is called *regularization.* Regularization is any modification of the ML algorithms that functions to reduce its generalization error, but not its training error (Goodfellow et al., 2016).

MORE ELABORATE ARCHITECTURES—DL NEURAL NETWORKS

The neural networks in the human brain have many more than just three layers, in which many of the layers are built from sensory input patterns formed by many different experiences. This means that different variable data are input to different layers, each one the result of different experiences. The encapsulation of this experience-based component of human neural networks was simulated by adding additional layers to the ANN architecture, fed by different sets of data inputs, analogous to different experiences. The resulting ANN architecture was far "deeper" in terms of hidden layers and data inputs than the original MLP, and was termed a deep learning neural net (DLNN). One of the early applications of deep learning neural networks (DLNNs) was to build the IBM Deep Blue chess-playing computer system, which beat the reigning world champion chess player, Gary Kasparov, in 1997 (Schmidhuber, 2015).

Major elaborations of DLNNs include

1. representational learning elements,
2. convolutions,
3. separate date sets input in some hidden nodes.

Representational Learning

An earlier development of learning machines involved mapping raw inputs to transformed features that "represented" raw data inputs. Analysis of these transformed inputs was called "representational learning" (RL). The problem with representational learning (RL) applied to early speech and image analysis was that it was difficult to extract nuances like speech accent or viewing angle of an image. DLNNs solve this problem in RL by using a group of simple representations to build more complex representations.

One application of a DLNN for image recognition dedicates the processing of one hidden layer to distinguish image elements contours, and the second hidden layer to distinguish edges, and the third hidden layer to combine contours and edges to compose image parts (Schmidhuber, 2015).

Convolutional Neural Networks

One of the earliest applications of complex ANNs (those with more than three layers) was image processing. Studies in computational neuroscience of vision (initially of a cat) suggested that a subtle change in the mathematical operations of a machine-learning algorithm could have profound benefits on both computational efficiency and noise reduction. The change was to use one function to modify another function to make it work better. The basic idea was to use one function to estimate (or generalize) the state of another function, without having to solve the other function directly. Such a modification is called a *convolution*. For example, we might use an aggregating function to collect noisy inputs from a submarine sonar sensor, and calculate the average over a small interval of time. Rather than submitting the raw noisy signal to the processing programs, only the average is input. This approach can have very significant effects on lowering the processing time to generate the output, and the storage requirements for intermediate products. This averaging is called *pooling*. This example of sonar signal processing involves only one input, the signal; there are no predictor variables. Imagine how this approach could be leveraged to modify the processing of machine-learning algorithms with many predictor variable inputs. That is how a convolutional neural network (CNN) was conceived.

Neural network algorithm developers recognized that the mathematical processing (i.e., matrix multiplication operations) during image analysis could be highly optimized in each layer of the neural network by using various convolutions to estimate parameter values. These convolutional neural networks were very efficient and effective in the analysis of any data set that can be represented as a two-dimensional data structure (like pixels in an image). Some convolutional neural networks were developed to work with time-series data sets, which are essentially two-dimensional data structures (output dimension and the time dimension). Some specialized convolutional neural networks incorporated time delays in the input processing strategy, *time-delayed neural networks* (TDNNs).

In the early 1990s, AT&T developed convolutional neural networks (CNNs) for checking images in banks (Goodfellow et al., 2016). Subsequently, Nippon Electric Corporation (NEC) adapted this technology to read over 10% of the checks processed by them in the United States. Now, all checks are read with variations of this technology.

Sparsely Connected Neural Networks (SCNN)

Another aspect of neural net computing further developed into a DL process is sparsely connected neural networks (SCNNs). As the number of hidden layers increased, the amount of processing increased exponentially. A fully connected neural network associated a weight with the connection of each input neuron and hidden neuron. In a sparsely connected network, some of those connections are not made. Fig. 19.7 shows a fully connected neural network, and Fig. 19.8 shows its sparsely connected counterpart.

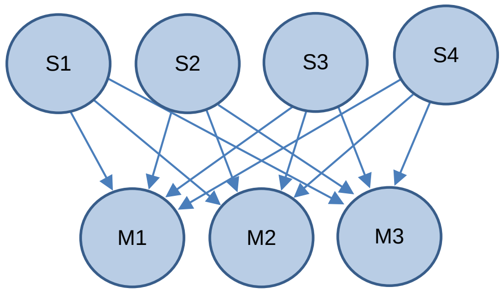

FIG. 19.7 A fully connected neural network.

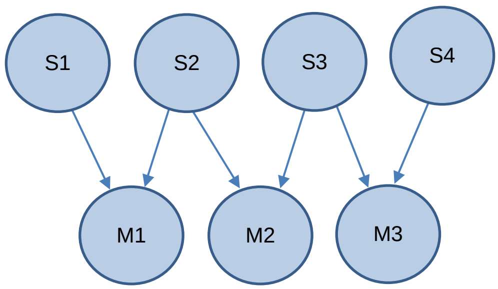

FIG. 19.8 A sparsely connected neural network (SCNN).

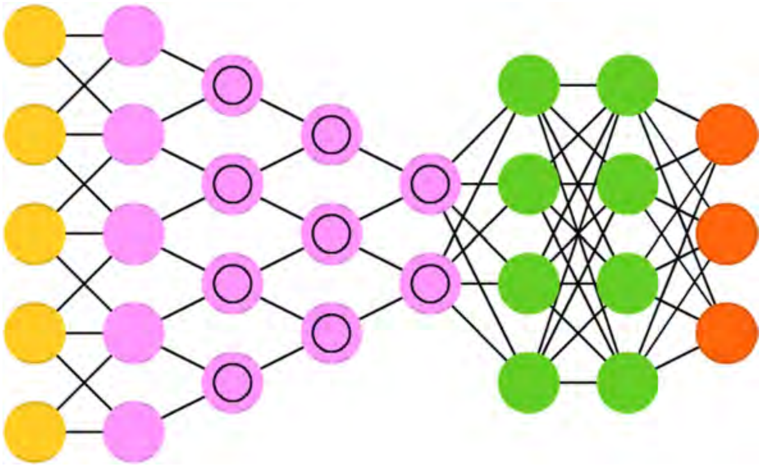

FIG. 19.9 A DLNN with fully connected and sparsely connected components. *From https://www.google.com/search?q=sparsely-connected+convoluted+neural+network+images&biw=1536&bih=735&tbm= isch&imgil=yoWhU67BLz_rKM%253A%253BigBynsCt8wHKMM%253Bhttps%25253A%25252F%25252Fwww.anyline.io%25252Fblog%25252Ftag%25252Fdeep-learning%25252F&source=iu&pf=m&fir=yoWhU67BLz_rKM% 253A%252CigBynsCt8wHKMM%252C_&usg=__-j4KI_DzX8EFkphETaRg1wIuyvI%3D&dpr=1.25&ved= 0ahUKEwi466KJ8q7SAhXMPiYKHcjTCkIQyjcIVg&ei=aWGzW-PiSMcz9mAHIp6uQBA#imgrc=IxHDl54iCmB1xM.*

The sparsely connected neural network (SCNN) are a lot less "noisy," yet it may capture enough of the pattern to be very useful. It requires much less processing, particularly for many records and many variables. Sparse connectivity is a feature of many CNNs.

In many DLNN algorithms, the fully connected design is combined with the sparsely connected design in the same network architecture (Fig. 19.9).

Notice in Fig. 19.9 that middle layer neurons in last three layers are *fully connected* with the layer to the left in the image and the output layer to the right, whereas neurons in the previous layers are *sparsely connected* (not connected to each neuron in the layers next to them).

Recurrent ANNs

Another kind of neural network that may be incorporated into DL systems is the recurrent neural network (RNN). Recurrent neural networks (RNNs) are very different from CNNs in the ways they can analyze temporal data inputs and generate sequential data output (Vorhies, 2016).

In contract to the standard feed-forward ANNs, RNNs have bidirectional data flow. The standard feed-forward data flow occurs, followed sometimes by feed-backward of data processed in later steps to affect the processing of earlier steps. This processing is referred to as back-propagation through time. This design does not make the assumption of other ANNs that data inputs are independent of each other in their effect on the output.

Temporal data. RNNs were modified subsequently to process various types of temporal data:

- Blocks of text if different lengths
- Audio speech signals

- Sequences of stock prices
- Streams of sensor data

Each of these types of temporal data can be viewed as sequential data through time. RNNs can learn from any data expressed as a sequence of data elements.

Memory. Not only can the feed-backward design in RNNs loop just one data unit backward in time, but also it can "look" backward many time units in the past. This feature permits simulation of learning over a timescale specified by the user. Proper tuning of the algorithm can control how long it "remembers" and when to "forget."

This ability to regulate when to remember and when to forget can help to solve an underlying problem with RNNs. Each time step in the processing of an RNN is equivalent to training a feed-forward ANN with 100 hidden layers. This situation leads to very small gradients in the error surface over which the gradient descent error minimization process must operate. As the number of time steps increases, the extent of the error surface declines exponentially, and it is known as the "vanishing gradient problem." The most common technique is to use the long short-term memory (LSTM) approach. This feature is particularly useful with sensor data that have long delays, or when there is a mixture of high and low frequency data (Vorhies, 2016).

Vorhies (2016) lists a number of applications for which RNNs are useful:

- Speech recognition
- Text processing
- Handwriting recognition
- Image recognition
- Supply-chain demand forecasting

Multiple Input Data Sets

Sometimes, different data sets are input to each hidden layer. These intermediate inputs are analogous to a serial set of inputs to human neural networks from different experiences, which drive the sequential learning processes.

Fig. 19.10 shows one way of relating DL to allied disciplines of representational learning, AI, and machine-learning.

POSTSCRIPT

Now, the groundwork has been laid upon which can be presented as the current focus of much of the DL technology—the development of the IBM Watson computer, described in Chapter 22. Before we do that, however, we must consider two caveats in the use of DL technology.

1. We must relate the concept of prediction accuracy in machine-learning methods to the concept of significance in parametric statistical analysis (Chapter 20).
2. We must consider the ethical aspects of the application of knowledge gained from predictive analytics studies (Chapter 21).

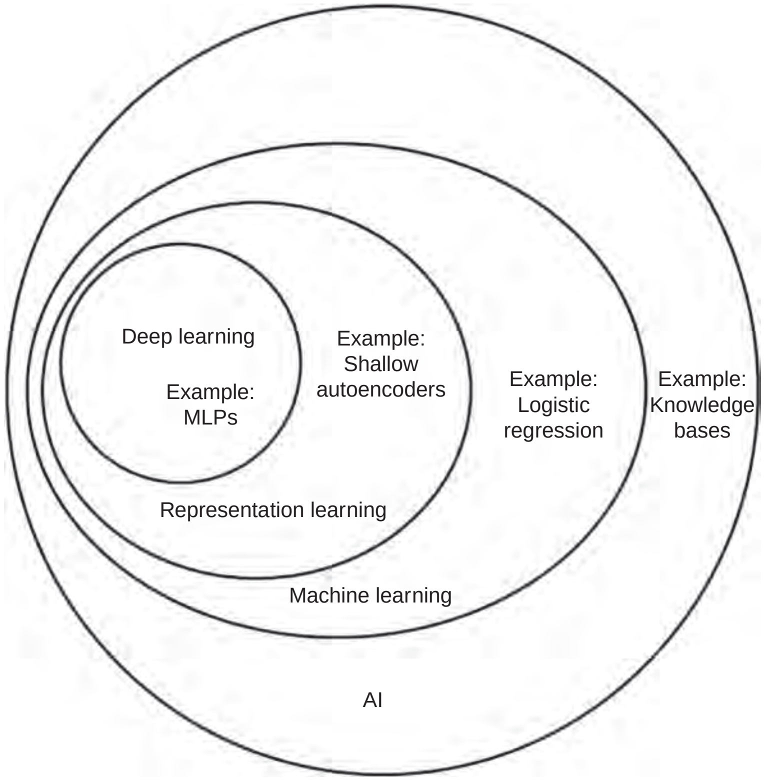

FIG. 19.10 Venn diagram of the relationships between the four levels of AI. *From Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT Press. http://www.deeplearningbook.org (in preparation).*

References

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. 800 pp.

Price, D., 1974. Gears from the Greeks. The Antikythera Mechanism: a calendar computer from ca. 80 B.C. Trans. Am. Philos. Soc. 64 (7), 1–70. https://doi.org/10.2307/1006146.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65 (3), 386–408.

Rosenblatt, F., 1961. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington, DC.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Netw. 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

Vorhies, W., 2016. Recurrent neural nets—the third and least appreciated leg of the AI stool. Online post. Available from: http://www.datasciencecentral.com/profiles/blogs/recurrent-neural-nets-the-third-and-least-appreciated-leg-of-the.

Further Reading

Garson, G.D., 1998. Neural Networks: An Introductory Guide for Social Scientists. Sage, London.