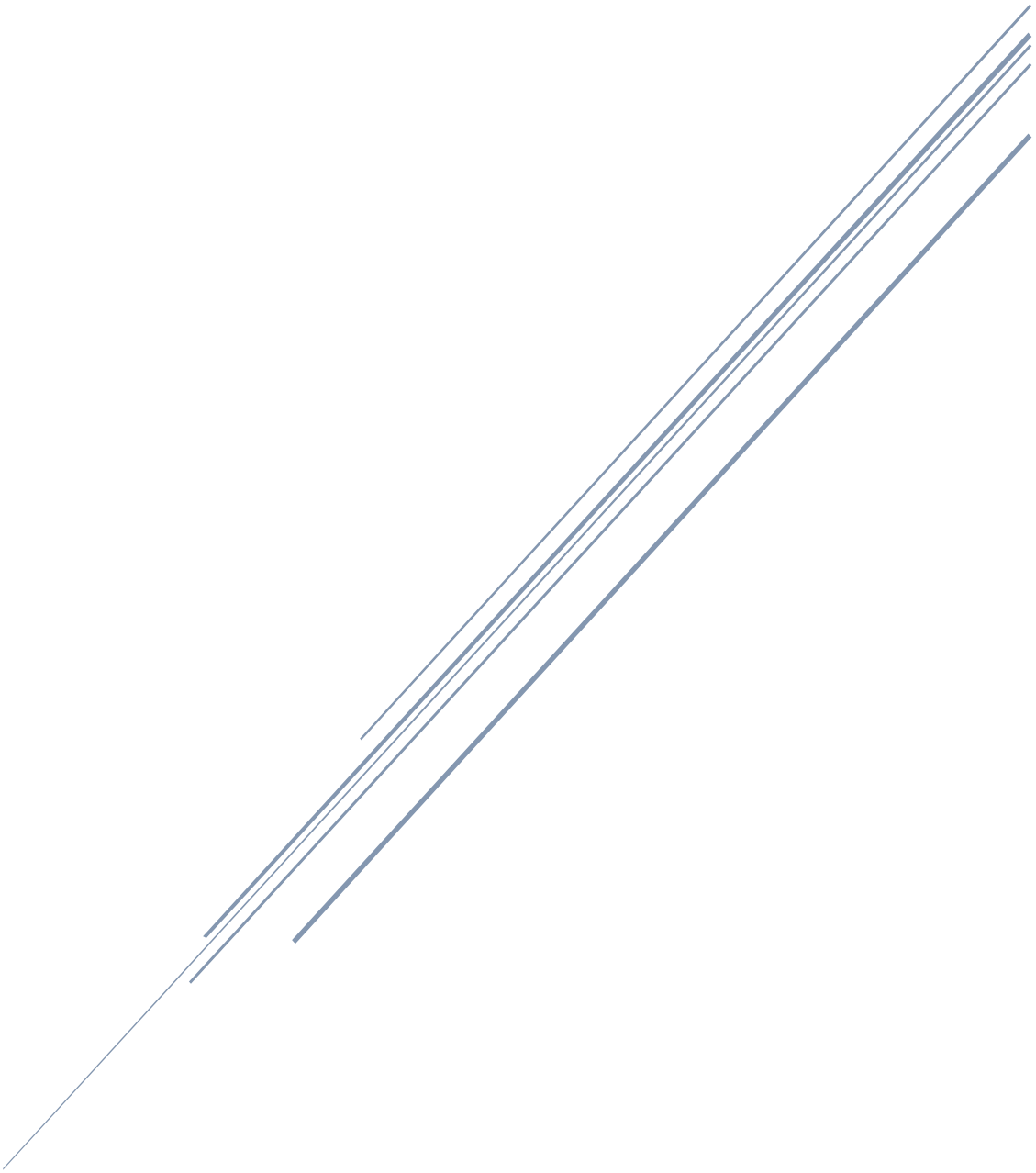
CHAPTER 2

Introduction to Machine Learning



2.1 Machine learning methods

Machine learning methods can broadly be divided into three categories: supervised, unsupervised, and reinforcement learning. Today, state-of-the-art techniques in all three categories rely on deep learning (Figure 2.1).

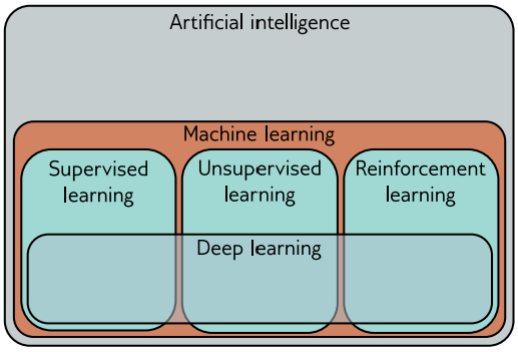


Figure 2.1 Machine learning is an area of artificial intelligence that fits mathematical models to observed data. It can coarsely be divided into supervised learning, unsupervised learning, and reinforcement learning. Deep neural networks contribute to each of these areas.

2.1.1 Supervised learning

Supervised learning models define a mapping from input data to an output prediction.

2.1.1.1 Regression and classification problems

Figure 2.2 depicts several regression and classification problems. In each case, there is a meaningful real-world input (a sentence, a sound file, an image, etc.), and this is encoded as a vector of numbers. This vector forms the model input. The model maps the input to an output vector which is then "translated" back to a meaningful real-world prediction.

The model in figure 2.2a predicts the price of a house based on input characteristics such as the square footage and the number of bedrooms. This is a regression problem because the model returns a continuous number (rather than a category assignment). In contrast, the model in figure 2.2b takes the chemical structure of a molecule as an input and predicts both the freezing and boiling points. This is a multivariate regression problem since it predicts more than one number. The model in figure 2.2c receives a text string containing a restaurant review as input and predicts whether the review is positive or negative. This is a binary classification problem because the model attempts to assign the input to one of two categories. The output vector contains the probabilities that the input belongs to each category. Figures 2.2d and 2.2e depict multiclass classification problems. Here, the model assigns the input to one of $N > 2$ categories. In the first case, the input is an audio file, and the model predicts which genre of music it contains. In the second case, the input is an image, and the model predicts which object it contains. In each case, the model returns a vector of size N that contains the probabilities of the N categories.

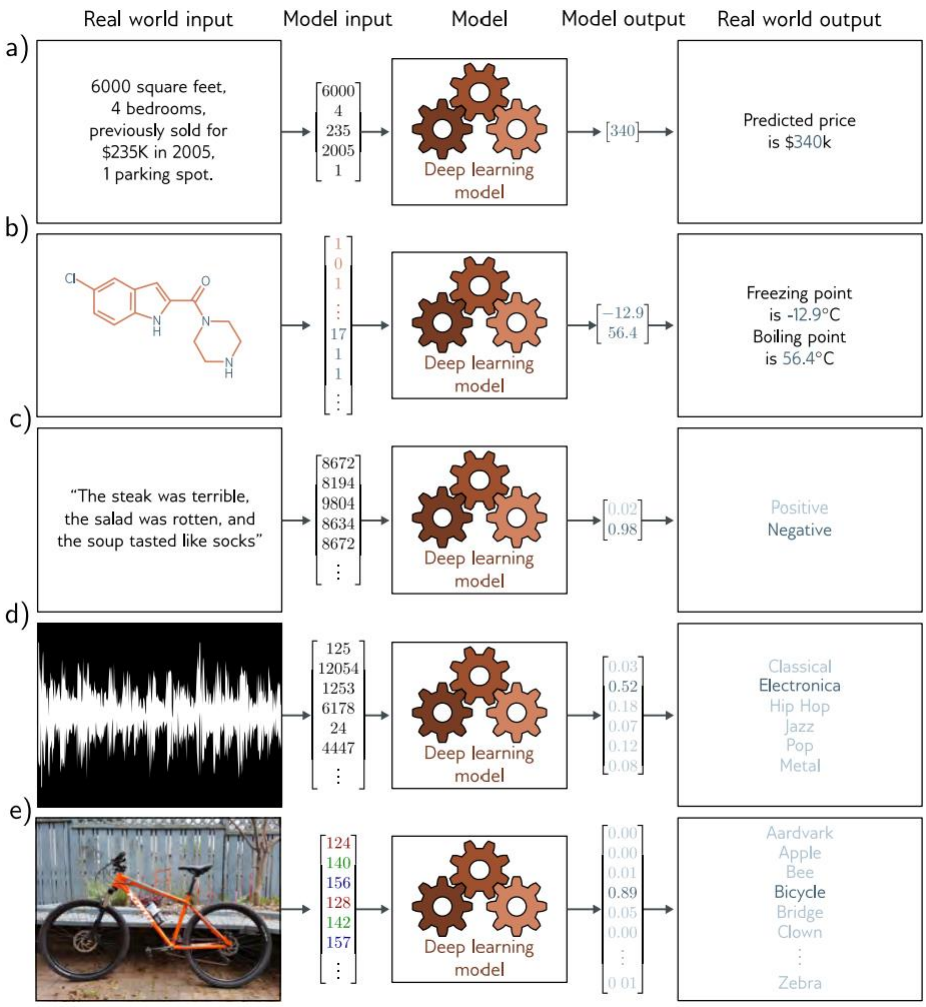


Figure 2.2 Regression and classification problems.

2.1.1.2 Inputs

The input data in figure 2.2 varies widely. In the house pricing example, the input is a fixed-length vector containing values that characterize the property. This is an example of tabular data because it has no internal structure; if we change the order of the inputs and build a new model, then we expect the model prediction to remain the same.

Conversely, the input in the restaurant review example is a body of text. This may be of variable length depending on the number of words in the review, and here input order is important; *my wife ate the chicken* is not the same as *the chicken ate my wife*. The text must be encoded into numerical form before passing it to the model. Here, we use a fixed vocabulary of size 10,000 and simply concatenate the word indices.

For the music classification example, the input vector might be of fixed size (perhaps a 10-second clip) but is very high-dimensional (i.e., contains many entries). Digital audio is usually sampled at 44.1 kHz and represented by 16-bit integers, so a ten-second clip consists of 441, 000 integers. Clearly, supervised learning models will have to be able to process sizeable inputs. The input in the image classification

example (which consists of the concatenated RGB values at every pixel) is also enormous. Moreover, it contains spatial structure; two pixels above and below one another are closely related, even if they are not adjacent in the input vector.

Finally, consider the input for the model that predicts the freezing and boiling points of the molecule. A molecule may contain varying numbers of atoms that can be connected in different ways. In this case, the model must ingest both the geometric structure of the molecule and the constituent atoms to the model.

2.1.1.3 Structured outputs

Figure 2.3a depicts a multivariate binary classification model for semantic segmentation. Here, every pixel of an input image is assigned a binary label that indicates whether it belongs to a cow or the background. Figure 2.3b shows a multivariate regression model where the input is an image of a street scene and the output is the depth at each pixel. In both cases, the output is high-dimensional and structured. However, this structure is closely tied to the input, and this can be exploited; if a pixel is labeled as "cow," then a neighbor with a similar RGB value probably has the same label.

Figures 2.3c–e depict three models where the output has a complex structure that is not so closely tied to the input. Figure 2.3c shows a model where the input is an audio file and the output is the transcribed words from that file. Figure 2.3d is a translation model in which the input is a body of text in English, and the output contains the French translation. Figure 2.3e depicts a very challenging task in which the input is descriptive text, and the model must produce an image that matches this description.

In principle, the latter three tasks can be tackled in the standard supervised learning framework, but they are more difficult for two reasons. First, the output may genuinely be ambiguous; there are multiple valid translations from an English sentence to a French one and multiple images that are compatible with any caption. Second, the output contains considerable structure; not all strings of words make valid English and French sentences, and not all collections of RGB values make plausible images. In addition to learning the mapping, we also have to respect the "grammar" of the output. Fortunately, this "grammar" can be learned without the need for output labels. For example, we can learn how to form valid English sentences by learning the statistics of a large corpus of text data. This provides a connection with the next section, which considers unsupervised learning models.

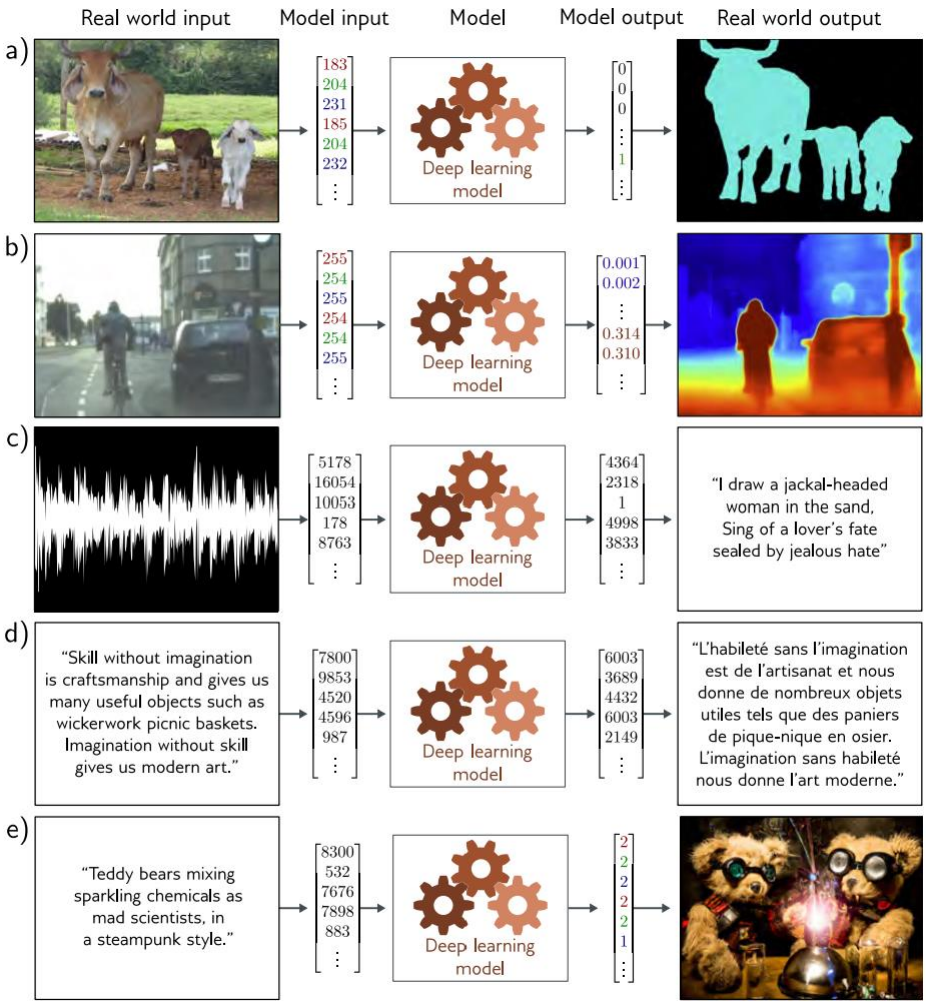


Figure 2.3 Supervised learning tasks with structured outputs.

2.1.2 Unsupervised learning

Constructing a model from input data without corresponding output labels is termed unsupervised learning; the absence of output labels means there can be no "supervision." Rather than learning a mapping from input to output, the goal is to describe or understand the structure of the data. As was the case for supervised learning, the data may have very different characteristics; it may be discrete or continuous, low-dimensional or high-dimensional, and of constant or variable length.

2.1.2.1 Generative models

Generative unsupervised models learn to synthesize new data examples that are statistically indistinguishable from the training data. Some generative models explicitly describe the probability distribution over the input data and here new examples are generated by sampling from this distribution. For example, Variational Autoencoders (VAEs). Others merely learn a mechanism to generate new examples without explicitly describing their distribution, such as Generative Adversarial Networks (GANs), Diffusion Models (e.g., DALL·E 2, Stable Diffusion), Transformers for Generation (e.g., GPT, BERT for masked token prediction). State-of-the-art generative models can synthesize examples that are extremely plausible but distinct from the training examples. They have been particularly successful at generating images (figure 2.4) and text (figure 2.5). They can also synthesize data under the constraint

that some outputs are predetermined (termed conditional generation). Examples include image inpainting (figure 2.6) and text completion (figure 2.7). Indeed, modern generative models for text are so powerful that they can appear intelligent. Given a body of text followed by a question, the model can often "fill in" the missing answer by generating the most likely completion of the document. However, in reality, the model only knows about the statistics of language and does not understand the significance of its answers.

Figure 2.4 Generative models for images. Left: two images were generated from a model trained on pictures of cats. These are not real cats, but samples from a probability model. Right: two images generated from a model trained on images of buildings.

The moon had risen by the time I reached the edge of the forest, and the light that filtered through the trees was silver and cold. I shivered, though I was not cold, and quickened my pace. I had never been so far from the village before, and I was not sure what to expect. I had been walking for hours, and I was tired and hungry. I had left in such a hurry that I had not thought to pack any food, and I had not thought to bring a weapon. I was unarmed and alone in a strange place, and I did not know what I was doing.

I had been walking for so long that I had lost all sense of time, and I had no idea how far I had come. I only knew that I had to keep going. I had to find her. I was getting close. I could feel it. She was nearby, and she was in trouble. I had to find her and help her, before it was too late.

Figure 2.5 Short story synthesized from a generative model of text data. The model describes a probability distribution that assigns a probability to every output string. Sampling from the model creates strings that follow the statistics of the training data (here, short stories) but have never been seen before.

Figure 2.6 Inpainting. In the original image (left), the boy is obscured by metal cables. These undesirable regions (center) are removed and the generative model synthesizes a new image (right) under the constraint that the remaining pixels must stay the same.

I was a little nervous before my first lecture at the University of Bath. It seemed like there were hundreds of students and they looked intimidating. I stepped up to the lectern and was about to speak when something bizarre happened.

Suddenly, the room was filled with a deafening noise, like a giant roar. It was so loud that I couldn't hear anything else and I had to cover my ears. I could see the students looking around, confused and frightened. Then, as quickly as it had started, the noise stopped and the room was silent again.

I stood there for a few moments, trying to make sense of what had just happened. Then I realized that the students were all staring at me, waiting for me to say something. I tried to think of something witty or clever to say, but my mind was blank. So I just said, "Well, that was strange,' and then I started my lecture.

Figure 2.7 Conditional text synthesis. Given an initial body of text (in black), generative models of text can continue the string plausibly by synthesizing the "missing" remaining part of the string.

2.1.2.2 Latent variables

Some (but not all) generative models exploit the fact that data can be lower dimensional than the raw number of observed variables suggests. For example, the number of valid and meaningful English sentences is much smaller than the number of strings created by drawing words at random. Similarly, real-world images are a tiny subset of the images that can be created by drawing random red, green, and blue (RGB) values for every pixel. This is because images are generated by physical processes (see figure 2.8).

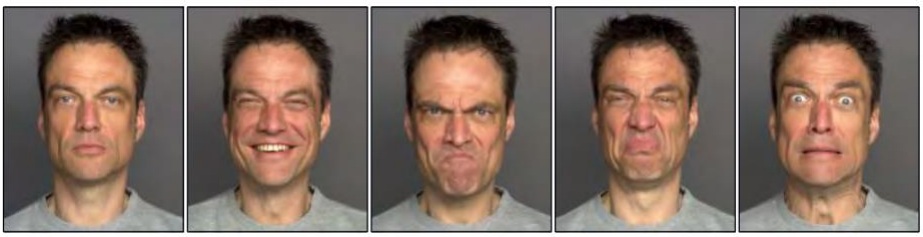


Figure 2.8 Variation of the human face. The human face contains roughly 42 muscles, so it's possible to describe most of the variation in images of the same person in the same lighting with just 42 numbers. In general, datasets of images, music, and text can be described by a relatively small number of underlying variables although it is typically more difficult to tie these to particular physical mechanisms.

This leads to the idea that we can describe each data example using a smaller number of underlying latent variables. Here, the role of deep learning is to describe the mapping between these latent variables and the data. The latent variables typically have a simple probability distribution by design. By sampling from this distribution and passing the result through the deep learning model, we can create new samples (figure 2.9). These models lead to new methods for manipulating real data. For example, consider finding the latent variables that underpin two real examples. We can interpolate between these examples by interpolating between their latent representations and mapping the intermediate positions back into the data space (figure 2.10).

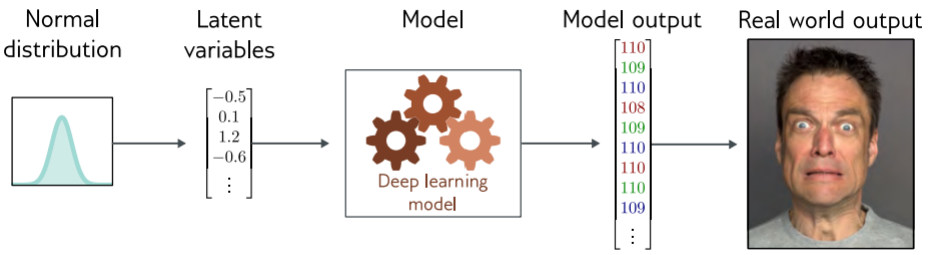


Figure 2.9 Latent variables. Many generative models use a deep learning model to describe the relationship between a low-dimensional "latent" variable and the observed high-dimensional data. The latent variables have a simple probability distribution by design. Hence, new examples can be generated by sampling from the simple distribution over the latent variables and then using the deep learning model to map the sample to the observed data space.

Figure 2.10 Image interpolation. In each row the left and right images are real and the three images in between represent a sequence of interpolations created by a generative model. The generative models that underpin these interpolations have learned that all images can be created by a set of underlying latent variables. By finding these variables for the two real images, interpolating their values, and then using these intermediate variables to create new images, we can generate intermediate results that are both visually plausible and mix the characteristics of the two original images.

2.1.2.3 Connecting supervised and unsupervised learning

Generative models with latent variables can also benefit supervised learning models where the outputs have structure (figure 2.3). For example, consider learning to predict the images corresponding to a caption. Rather than directly map the text input to an image, we can learn a relation between latent variables that explain the text and the latent variables that explain the image. This has three advantages. First, we may need fewer text/image pairs to learn this mapping now that the inputs and outputs are lower dimensional. Second, we are more likely to generate a plausible-looking image; any sensible values of the latent variables should produce something that looks like a plausible example. Third, if we introduce randomness to either the mapping between the two sets of latent variables or the mapping from the latent variables to the image, then we can generate multiple images that are all described well by the caption (figure 2.11).

Figure 2.11 Multiple images generated from the caption "A teddy bear on a skateboard in Times Square." Generated by DALL·E-2.

2.1.3 Reinforcement learning

The final area of machine learning is reinforcement learning. This paradigm introduces the idea of an agent which lives in a world and can perform certain actions at each time step. The actions change the state of the system but not necessarily in a deterministic way. Taking an action can also produce rewards, and the goal of reinforcement learning is for the agent to learn to choose actions that lead to high rewards on average. One complication is that the reward may occur sometime after the action is taken, so associating a reward with an action is not straightforward. This is known as the *temporal credit assignment problem*. As the agent learns, it must trade off exploration and exploitation of what it already knows; perhaps the agent has already learned how to receive modest rewards; should it follow this strategy (exploit what it knows), or should it try different actions to see if it can improve (explore other opportunities)?

2.1.3.1 Two examples

Consider teaching a humanoid robot to locomote. The robot can perform a limited number of actions at a given time (moving various joints), and these change the state of the world (its pose). We might reward the robot for reaching checkpoints in an obstacle course. To reach each checkpoint, it must perform many actions, and it's unclear which ones contributed to the reward when it is received and which were irrelevant. This is an example of the temporal credit assignment problem. A second example is learning to play chess. Again, the agent has a set of valid actions (chess moves) at any given time. However, these actions change the state of the system in a non-deterministic way; for any choice of action, the opposing player might respond with many different moves. Here, we might set up a reward structure based on capturing pieces or just have a single reward at the end of the game for winning. In the latter case, the temporal credit assignment problem is extreme; the system must learn which of the many moves it made were instrumental to success or failure. The exploration-exploitation trade-off is also apparent in these two examples. The robot may have discovered that it can make progress by lying on its side and pushing with one leg. This strategy will move the robot and yields rewards, but much more slowly than the optimal solution: to balance on its legs and walk. So, it faces a choice between exploiting what it already knows (how to slide along the floor awkwardly) and exploring the space of actions (which might result in much faster locomotion). Similarly, in the chess example, the agent may learn a reasonable sequence of opening moves. Should it exploit this knowledge or explore different opening sequences? It is perhaps not obvious how deep learning fits into the reinforcement learning framework. There are several possible approaches, but one technique is to use deep networks to build a mapping from the observed world state to an action. This is known as a policy network. In the robot example, the policy network would learn a mapping from its sensor measurements to joint movements. In the

chess example, the network would learn a mapping from the current state of the board to the choice of move (figure 2.13).

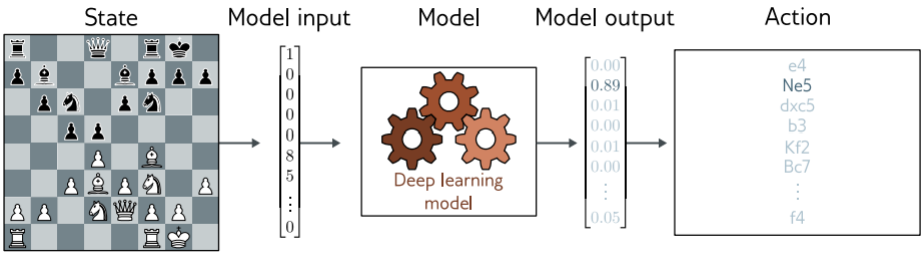


Figure 2.13 Policy networks for reinforcement learning. One way to incorporate deep neural networks into reinforcement learning is to use them to define a mapping from the state (here position on chessboard) to the actions (possible moves). This mapping is known as a policy.