The MIT Press Essential Knowledge Series

A complete list of the titles in this series appears at the back of this book.

DEEP LEARNING

JOHN D. KELLEHER

The MIT Press | Cambridge, Massachusetts | London, England

This book was set in Chaparral Pro by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Names: Kelleher, John D., 1974- author.
Title: Deep learning / John D. Kelleher.
Description: Cambridge, MA : The MIT Press, [2019] | Series: The MIT press essential knowledge series | Includes bibliographical references and index.
Identifiers: LCCN 2018059550 | ISBN 9780262537551 (pbk. : alk. paper)
Subjects: LCSH: Machine learning. | Artificial intelligence.
Classification: LCC Q325.5 .K454 2019 | DDC 006.3/1—dc23 LC record available at https://lccn.loc.gov/2018059550

10 9 8 7 6 5

CONTENTS

the network using large datasets. By contrast, hyperparameters are the parameters of a model (in these cases, the parameters of a neural network architecture) and/or training algorithm that cannot be directly estimated from the data but instead must be specified by the person creating the model, either through the use of heuristic rules, intuition, or trial and error. Often, much of the effort that goes into the creation of a deep learning network involves experimental work to answer the questions in relation to hyperparameters, and this process is known as hyperparameter tuning. The next chapter will review the history and evolution of deep learning, and the challenges posed by many of these questions are themes running through the review. Subsequent chapters in the book will explore how answering these questions in different ways can create networks with very different characteristics, each suited to different types of tasks. For example, recurrent neural networks are best suited to processing sequential/time-series data, whereas convolutional neural networks were originally developed to process images. Both of these network types are, however, built using the same fundamental processing unit, the artificial neuron; the differences in the behavior and abilities of these networks stems from how these neurons are arranged and composed.

4

A BRIEF HISTORY OF DEEP LEARNING

The history of deep learning can be described as three major periods of excitement and innovation, interspersed with periods of disillusionment. Figure 4.1 shows a timeline of this history, which highlights these periods of major research: on threshold logic units (early 1940s to the mid 1960s), connectionism (early 1980s to mid-1990s), and deep learning (mid 2000s to the present). Figure 4.1 distinguishes some of the primary characteristics of the networks developed in each of these three periods. The changes in these network characteristics highlight some of the major themes within the evolution of deep learning, including: the shift from binary to continuous values; the move from threshold activation functions, to logistic and tanh activation, and then onto ReLU activation; and the progressive deepening of the networks, from single layer, this time that the term deep learning came to prominence to describe deep neural networks. The term deep learning is used to emphasize the fact that the networks being trained are much deeper than previous networks.

One of the early successes of this new era of neural network research was when Geoffrey Hinton and his colleagues demonstrated that it was possible to train a deep neural network using a process known as greedy layer-wise pretraining. Greedy layer-wise pretraining begins by training a single layer of neurons that receives input directly from the raw input. There are a number of different ways that this single layer of neurons can be trained, but one popular way is to use an autoencoder. An autoencoder is a neural network with three layers: an input layer, a hidden (encoding) layer, and an output (decoding) layer. The network is trained to reconstruct the inputs it receives in the output layer; in other words, the network is trained to output the exact same values that it received as input. A very important feature in these networks is that they are designed so that it is not possible for the network to simply copy the inputs to the outputs. For example, an autoencoder may have fewer neurons in the hidden layer than in the input and output layer. Because the autoencoder is trying to reconstruct the input at the output layer, the fact that the information from the input must pass through this bottleneck in the hidden layer forces the autoencoder to learn an encoding of the input data in the

hidden layer that captures only the most important features in the input, and disregards redundant or superfluous information.[7]

Layer-Wise Pretraining Using Autoencoders

In layer-wise pretraining, the initial autoencoder learns an encoding for the raw inputs to the network. Once this encoding has been learned, the units in the hidden encoding layer are fixed, and the output (decoding) layer is thrown away. Then a second autoencoder is trained—but this autoencoder is trained to reconstruct the representation of the data generated by passing it through the encoding layer of the initial autoencoder. In effect, this second autoencoder is stacked on top of the encoding layer of the first autoencoder. This stacking of encoding layers is considered to be a greedy process because each encoding layer is optimized independently of the later layers; in other words, each autoencoder focuses on finding the best solution for its immediate task (learning a useful encoding for the data it must reconstruct) rather than trying to find a solution to the overall problem for the network.

Once a sufficient number[8] of encoding layers have been trained, a tuning phase can be applied. In the tuning phase, a final network layer is trained to predict the target output for the network. Unlike the pretraining of the earlier layers of the network, the target output for the final layer is different from the input vector and is specified in the training dataset. The simplest tuning is where the pretrained layers are kept frozen (i.e., the weights in the pretrained layers don't change during the tuning); however, it is also feasible to train the entire network during the tuning phase. If the entire network is trained during tuning, then the layer-wise pretraining is best understood as finding useful initial weights for the earlier layers in the network. Also, it is not necessary that the final prediction model that is trained during tuning be a neural network. It is quite possible to take the representations of the data generated by the layer-wise pretraining and use it as the input representation for a completely different type of machine learning algorithm, for example, a support vector machine or a nearest neighbor algorithm. This scenario is a very transparent example of how neural networks learn useful representations of data prior to the final prediction task being learned. Strictly speaking, the term pretraining describes only the layer-wise training of the autoencoders; however, the term is often used to refer to both the layer-wise training stage and the tuning stage of the model.

Figure 4.5 shows the stages in layer-wise pretraining. The figure on the left illustrates the training of the initial autoencoder where an encoding layer (the black circles) of three units is attempting to learn a useful representation for the task of reconstructing an input vector of length 4. The figure in the middle of figure 4.5 shows the training of a second autoencoder stacked on top of the encoding layer

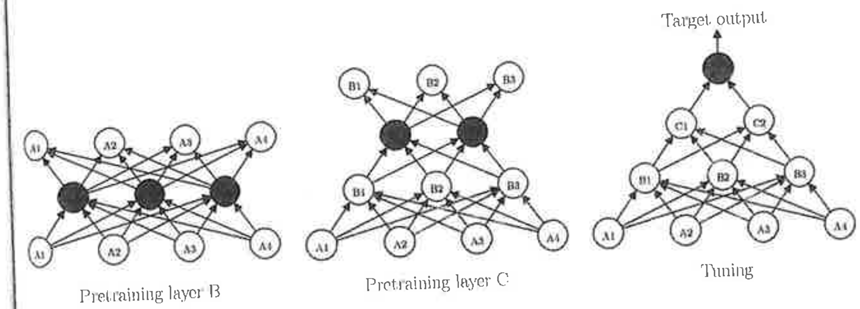

Figure 4.5 The pretraining and tuning stages in greedy layer-wise pretraining. Black circles represent the neurons whose training is the primary objective at each training stage. The gray background marks the components in the network that are frozen during each training stage.

of the first autoencoder. In this autoencoder, a hidden layer of two units is attempting to learn an encoding for an input vector of length 3 (which in turn is an encoding of a vector of length 4). The grey background in each figure demarcates the components in the network that are frozen during this training stage. The figure on the right shows the tuning phase where a final output layer is trained to predict the target feature for the model. For this example, in the tuning phase the pretrained layers in the network have been frozen.

Layer-wise pretraining was important in the evolution of deep learning because it was the first approach to training deep networks that was widely adopted.[9] However, today most deep learning networks are trained without using layer-wise pretraining. In the mid-2000s, researchers began to appreciate that the vanishing gradient problem was not a strict theoretical limit, but was instead a practical obstacle that could be overcome. The vanishing gradient problem does not cause the error gradients to disappear entirely; there are still gradients being backpropagated through the early layers of the network, it is just that they are very small. Today, there are a number of factors that have been identified as important in successfully training a deep network.

Weight Initialization and ReLU Activation Functions

One factor that is important in successfully training a deep network is how the network weights are initialized. The principles controlling how weight initialization affects the training of a network are still not clear. There are, however, weight initialization procedures that have been empirically shown to help with training a deep network. Glorot initialization[10] is a frequently used weight initialization procedure for deep networks. It is based on a number of assumptions but has empirical success to support its use. To get an intuitive understanding of Glorot initialization, consider the fact that there is typically a relationship between the magnitude of values in a set and the variance of the set: generally the larger the values in a set, the larger the variance of the set. So, if the variance calculated on a set of gradients propagated through a layer

In the mid-2000s, researchers began to appreciate that the vanishing gradient problem was not a strict theoretical limit, but was instead a practical obstacle that could be overcome.