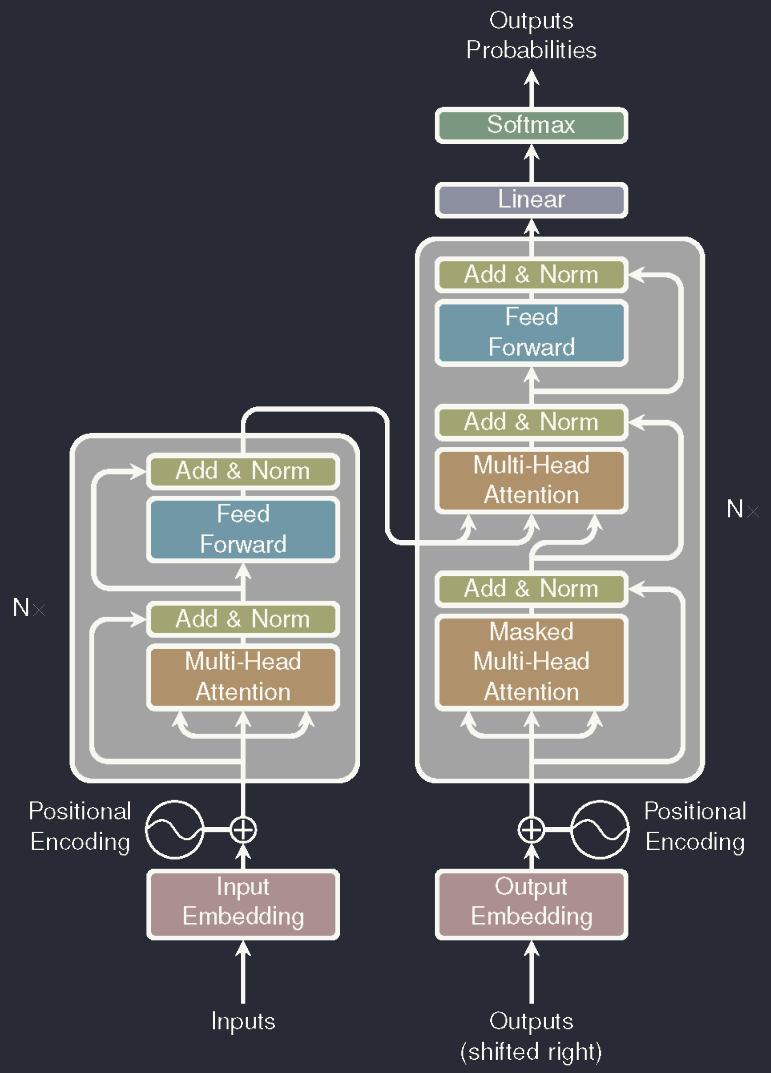

Figure 1: "The Transformer" Recreated in TikZ from "Attention Is All You Need"

Machine Learning Notes (Book 1)

"Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Author

Paul Beggs

PaulBeggs03@gmail.com

ACKNOWLEDGEMENTS AND ROADMAP

Thank you to *Aurélien Géron* for writing the book *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. This document is a compilation of notes taken while reading the book. The notes are not exhaustive and are meant to be a reference for myself.

This book is organized in two parts. Part I, "The Fundamentals of Machine Learning", covers the following topics:

- What machine learning is, what problems it tries to solve, and the main categories and fundamental concepts of its systems
- The steps in a typical machine learning project
- Learning by fitting a model to data
- Optimizing a cost function
- Handling, cleaning, and preparing data
- Selecting and engineering features
- Selecting a model and tuning hyperparameters using cross-validation
- The challenges of machine learning, in particular underfitting and overfitting (the bias/variance trade-off)
- The most common learning algorithms: linear and polynomial regression, logistic regression, k-nearest neighbors, support vector machines, decision trees, random forests, and ensemble methods
- Reducing the dimensionality of the training data to fight the "curse of dimensionality"
- Other unsupervised learning techniques, including clustering, density estimation, and anomaly detection

Part II, "Neural Networks and Deep Learning", covers the following topics:

- What neural nets are and what they're good for
- Building and training neural nets using TensorFlow and Keras

- The most important neural net architectures: feedforward neural nets for tabular data, convolutional nets for computer vision, recurrent nets and long short-term memory (LSTM) nets for sequence processing, encoder–decoders and transformers for natural language processing (and more!), autoencoders, generative adversarial networks (GANs), and diffusion models for generative learning
- Techniques for training deep neural nets
- How to build an agent (e.g., a bot in a game) that can learn good strategies through trial and error, using reinforcement learning
- Loading and preprocessing large amounts of data efficiently
- Training and deploying TensorFlow models at scale

The first part is based mostly on Scikit-Learn, while the second part uses TensorFlow and Keras.

TABLE OF CONTENTS

Part I

The Fundamentals of Machine Learning

CHAPTER 1

THE MACHINE LEARNING LANDSCAPE

1.1 Introduction

We will start by defining machine learning and why one would want to use it. Then, we will look at supervised versus unsupervised learning and their variants, online versus batch learning, and instance-based versus model-based learning. Then, we will look at the workflow of a typical ML project and discuss the main challenges and how to evaluate and fine-tune a machine learning system.

Machine Learning is the science (and art) of programming computers, so they can *learn* from data.

A more engineering based definition:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*
>
> —Tom Mitchell, 1997

1.2 Why Use Machine Learning?

Machine learning is good for the following main points:

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules (a machine learning model can often simplify code and perform better than the traditional approach)
- Complex problems for which using a traditional approach yields no good solution (the best machine learning techniques can perhaps find a solution)
- Fluctuating environments (a machine learning system can easily be retrained on new data, always keeping it up to date)
- Getting insights about complex problems and large amounts of data

1.2.1 Application Examples

Machine learning is used in many applications, including:

Analyzing images of products on a production line to automatically classify them

This is image classification, typically performed using convolutional neural networks (CNNs; see Chapter 14) or sometimes transformers (see Chapter 16).

Detecting tumors in brain scans

This is semantic image segmentation, where each pixel in the image is classified (as we want to determine the exact location and shape of tumors), typically using CNNs or transformers.

Automatically classifying news articles

This is natural language processing (NLP), and more specifically text classification, which can be tackled using recurrent neural networks (RNN), but transformers work even better (see Chapter 16).

Automatically flagging offensive comments on discussion forums

This is also text classification, using the same NLP tools.

Summarizing long documents automatically

This is a branch of NLP called text summarization, again using the same tools.

Creating a chatbot or personal assistant

This involves many NLP components, including natural language understanding (NLU) and question-answering modules.

Forecasting your company's revenue next year, based on many performance metrics

This is a regression task (i.e., predicting values) that may be tackled using any regression model, such as a linear regression or polynomial regression model (see Chapter 4), a regression support vector machine (see Chapter 5), a regression random forest (see Chapter 7), or an artificial neural network (see Chapter 10). If you want to take into account sequences of past performance metrics, you may want to use RNNs, CNNs, or transformers (see Chapters 15 and 16).

Marking your app react to voice commands

This is speech recognition, which requires processing audio samples: since they are long and complex sequences, they are typically pocessed using RNNs, CNNs, or transformers (see Chapters 15 and 16).

Detecting credit card fraud

This is anomaly detection, which can be tackled using isolation forests, Gaussian mixture models (see Chapter 9), or autoencoders (see Chapter 17).

Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment

This is clustering, which can be achieved using k-means, DBSCAN, and more (see Chapter 9).

Recommending a product that a client may be interested in, based on past purchases

This is a recommender system. One approach is to feed past purchases (and other information about the client) to an artificial neural network (see Chapter 10), and get it to output the most likely next purchase. This neural net would typically be trained on past sequences of purchases across all clients.

Building an intelligent bot for a game

This is often tackled using reinforcement learning (RL; see Chapter 18), which is a branch of machine learning that trains agents (Such as bots) to pick the actions that will maximize their rewards over time (e.g., a bot may get a reward every time the player loses some life points), within a given environment (such as the game). The famous AlphaGo program that beat the world champion at the game of Go was built using RL.

1.3 Types of Machine Learning Systems

Machine learning systems can be classified into several categories, depending on the type of data they are trained on and the type of tasks they are designed to perform. The main categories are:

- How they are supervised during training (supervised, unsupervised, or semi-supervised, self-supervised, and others)
- Whether or not they can learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model-based learning)

1.3.1 Training Supervision

This subsection goes over the main types of supervision used in machine learning: supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning.

Supervised Learning

Supervised Learning is when you feed an algorithm a training set of labeled examples, and the algorithm learns to predict the labels for new instances.

Classification a spam filter is a good example of this: it is trained with many example emails along with their *class* (spam or ham), and it must learn how to classify new emails.

Regression is when the labels are continuous values (e.g., predicting the price of a house). Training involves giving the model many examples of cars, including both their features and their targets (i.e., their prices).

NOTE: The words *target* and *label* are often used interchangeably, but *target* is more common in regression tasks, while *label* is more common in classification tasks. Moreover, *features* are sometimes called *predictors* or *attributes*. These terms may refer to individual samples (e.g., "this car's mileage feature is equal to 15,000"), or to all samples (e.g., "the mileage feature is strongly correlated with price.").

Unsupervised Learning

Unsupervised Learning is when you only feed the algorithm a training set of unlabeled examples, and the algorithm tries to learn the structure of the data.

Consider the example of running a website that collects data from shoppers. You can run a *clustering* algorithm on the data to group similar shoppers together. The algorithm will find connections between the shoppers without your assistance. For example, it might notice that 40% of shoppers are teenagers who buy candy on weekdays after school, and 20% are adults who only buy vegetables on the weekend. If you use *hierarchical clustering* you can subdivide these groups into smaller groups, such as "teenagers who buy candy on weekdays after school" and "teenagers who buy candy on weekends".

Visualization algorithms take complex, unlabeled data, and output a 2D or 3D representation that can easily be plotted. They preserve as much structure as possible (e.g., trying to keep separate clusters in the input space from overlapping in the visualization) so that you can perhaps identify unsuspected patterns.

A related task is *dimensionality reduction*, in which the goal is to simplify the data without losing too much information. You could merge several correlated features into one. For example, you could combine the features "height" and "weight" into a single feature called "body mass index (BMI)". This is called *feature extraction*.

TIP: Reducing the number of dimensions in training data by utilizing these dimensionality reduction algorithms will make the later machine learning algorithm run much faster, take up less memory and disk space, and in some cases, improve the performance of the algorithm.

Anomaly detection is a special case of unsupervised learning, where the goal is to identify rare items, events, or observations that raise suspicions by differing significantly from the majority of the data. For example, you could use anomaly detection to identify fraudulent credit card transactions.

Similarly, *novelty detection* is used to detect new instances that look different from all other instances in the training set. The caveat for using this method, is that your training set must be "clean" (i.e., devoid of any instance that you would like the algorithm to detect). For

example, if you have thousands of images of different dogs and 1% of these are Chihuahuas, then a novelty detection algorithm should not treat new pictures of Chihuahuas as novelties. On the other hand, anomaly detection algorithms may consider these dogs as so rare and different from other dogs that they would likely classify them as anomalies.

The final unsupervised task is *association rule learning*, which is used to discover interesting relations between variables in large databases. For example, suppose you own a supermarket. Running an association rule on your sales logs may reveal that people who purchase barbecue sauce and potato chips also tend to buy steak. Thus, you may want to place these items close to each other in your store.

Semi-Supervised Learning

Since labeling is often time-consuming and costly, some algorithms can deal with data that's partially labeled. This is called *semi-supervised learning*.

Google Photos uses semi-supervised learning to identify people in your photos. Once all your family photos are uploaded to the service, it automatically recognizes that the same person A shows up in photos 1, 5, and 11, while another person B shows up in photos 2, 5, and 7. This is the *unsupervised* part of the algorithm that utilizes clustering.

Most semi-supervised learning algorithms are combinations of unsupervised and supervised learning algorithms. For example, a clustering algorithm may group similar instances together, and then every unlabeled instance can be labeled with the most common label in its cluster. Once the whole dataset is labeled, a supervised learning algorithm can be trained on it.

Self-Supervised Learning