

PYTHON

MACHINE

LEARNING

THE ULTIMATE BEGINNER'S GUIDE TO LEARN PYTHON MACHINE
LEARNING STEP BY STEP USING SCIKIT-LEARN AND TENSORFLOW



RYAN TURNER

Python Machine Learning

*The Ultimate Beginner's Guide to
Learn Python Machine Learning Step
by Step using Scikit-Learn and
Tensorflow*

Ryan Turner

© Copyright 2019 – Ryan Turner - All rights reserved.

The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.

Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book. Either directly or indirectly.

Legal Notice:

This book is copyright protected. This book is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, and reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.

By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, — errors, omissions, or inaccuracies.

Table of Contents

Getting Started

[What is Machine Learning?](#)

[Classification of Machine Learning Algorithms](#)

[Supervised Learning](#)

[Unsupervised Learning](#)

[Reinforcement Learning](#)

[What is Deep Learning?](#)

[What is TensorFlow?](#)

Chapter 1: History of Machine Learning

Chapter 2: Theories of Machine Learning

Chapter 3: Approaches to Machine Learning

[Philosophies of Machine Learning](#)

[Supervised and Semi-supervised Learning Algorithms](#)

[Unsupervised Learning Algorithms](#)

[Reinforcement Learning](#)

Chapter 4: Environment Setup

[Installing Scikit-Learn](#)

[Installing TensorFlow](#)

Chapter 5: Using Scikit-Learn

[Loading Datasets](#)

[Regression](#)

Chapter 6: k-Nearest Neighbors Algorithm

[Splitting the Dataset](#)

[Feature Scaling](#)

[Training the Algorithm](#)

[Evaluating the Accuracy of the Algorithm](#)

[Comparing K Value with the Error Rate](#)

[Chapter 7: K-Means Clustering](#)

[Data Preparation](#)

[Visualizing the Data](#)

[Creating Clusters](#)

[Chapter 8: Support Vector Machines](#)

[Importing the Dataset](#)

[Preprocessing the Data](#)

[Training the Algorithm](#)

[Making Predictions](#)

[Evaluating the Accuracy of the Algorithm](#)

[Chapter 9: Machine Learning and Neural Networks](#)

[Feedforward Neural Networks](#)

[Recurrent Neural Networks](#)

[Chapter 10: Machine Learning and Big Data](#)

[Chapter 11: Machine Learning and Regression](#)

[Chapter 12: Machine Learning and the Cloud](#)

[Benefits of Cloud-Based Machine Learning](#)

[Chapter 13: Machine Learning and the Internet of Things \(IoT\)](#)

[Consumer Applications](#)

[Commercial Applications](#)

[Industrial Applications](#)

[Infrastructure Applications](#)

[Trends in IoT](#)

[Chapter 14: Machine Learning and Robotics](#)

[Examples of Industrial Robots and Machine Learning](#)

[Neural Networks with Scikit-learn](#)

[Chapter 15: Machine Learning and Swarm Intelligence](#)

[Swarm Behavior](#)

[Applications of Swarm Intelligence](#)

[Chapter 16: Machine Learning Models](#)

[Chapter 17: Applications of Machine Learning](#)

[Chapter 18: Programming and \(Free\) Datasets](#)

[Limitations of Machine Learning](#)

The Philosophical Objections: Jobs, Evil, and Taking Over the World
Chapter 19: Machine Learning and the Future
Conclusion

Getting Started

If you've worked with another type of programming, you're in luck. Translating programming languages is a cinch. Many programs use the same or similar codes and coding techniques. Not only does further learning and adaptations to various programming languages help you to understand machinery, but you will also be able to move to and from languages with veritable ease.

If you've never used Python before, it's common to experience some difficulties translating what you know into Python's language. Python, however, is one of the easiest forms of programming and is a great starter for anyone who wants to pick up on various languages. Quite often, Python can be found inexpensively or free, if you're willing to look for it.

Programming provides the means for electronics and various technology to operate and learn based on your behaviors. If you are familiar with devices such as Apple's Siri, Google Home, and others, you'll recognize that these devices are based on programming. Data acquisition through direct interaction with humans gives these devices the means to grow and enhance. If you understand this kind of programming or are eager to start learning how you can make and improve your own devices, continue reading. Now we'll concentrate on attaining the skills to work with programming on all levels.

What is Machine Learning?

The learning process begins with data or observations, like instruction, direct experience, or examples of extracting patterns from the data and using these patterns to make predictions in the future (Bose, 2019). The primary goal of machine learning is to allow computers to learn automatically without intervention by humans and adjust accordingly.

With machine learning, we can analyze large quantities of data. Machine learning gives us profitable results, but we may need a number of resources to reach this point. Additional time may be needed to train the machine learning models.

Classification of Machine Learning Algorithms

The machine learning algorithms can fall in the category of supervised, unsupervised, or reinforced learning.

Supervised Learning

For the case of supervised learning, the human is expected to provide both the inputs and the outputs which are desired and furnish the feedback based on the accuracy of the predictions during training. After completion of the training, the same algorithm will have to apply that was applied to the next data.

The concept of supervised learning can be seen as similar to learning under a teacher's supervision in human beings. The teacher gives some examples to the student, and the student then derives new rules and knowledge from these examples so as to apply this somewhere else.

It is also good for you to know the difference between the regression problems and classification problems. In regression problems, the target is a numeric value, while in classification, the target is a class or a tag. A regression task can help determine the average cost of all houses in London, while a classification task will help determine the types of flowers based on the length of their sepals and petals.

Unsupervised Learning

For the case of unsupervised learning, the algorithms do not expect to be provided with the output data. An approach called deep learning, which is an iterative approach, is used so as to review the data and arrive at new conclusions. This makes them suitable for use in processing tasks which are complex compared to the supervised learning algorithms. This means that the unsupervised learning algorithms learn solely from examples without responses to these. The algorithm finds patterns from the examples on its own.

Supervised learning algorithms work similarly to how humans determine any similarities between two or more objects. Majority of recommender systems you encounter when purchasing items online work based on unsupervised learning algorithms. In this case, the algorithm derives what to suggest to you to purchase from what you have purchased before. The algorithm has to estimate the kind of customers whom you resemble, and a suggestion is drawn from that.

Reinforcement Learning

This type of learning occurs when the algorithm is presented with examples which lack labels, as is the case with unsupervised learning. However, the example can be accompanied by positive or negative feedback depending on the solution which is proposed by the algorithm. It is associated with applications in which the algorithm has to make decisions, and these decisions are associated with a consequence. It is similar to trial and error in human learning.

Errors become useful in learning when they are associated with a penalty such as pain, cost, loss of time, etc. In reinforced learning, some actions are more likely to succeed compared to others.

Machine learning processes are similar to those of data mining and predictive modeling. In both cases, searching through the data is required so as to draw patterns and adjust the actions of the program accordingly. A good example of machine learning is the recommender systems. If you purchase an item online, you will get an ad which is related to that item, and that is a good example of machine learning.

What is Deep Learning?

It is through deep learning that a computer is able to learn to perform classification tasks directly from text, images, or sound. Deep learning models are able to achieve a state-of-the-art accuracy, which in some cases exceeds human-level performance. Large sets of labeled data and neural network architectures are used to train models in deep learning.

What is TensorFlow?

TensorFlow is a framework from Google used for the creation of deep learning models. TensorFlow relies on data-flow graphs for numerical computation. TensorFlow has made machine learning easy. It makes the processes of acquiring data, training machine learning models, making predictions, and modifying future results easy.

The library was developed by Google's Brain team for use in large-scale machine learning. TensorFlow brings together machine learning and deep learning algorithms and models, and it makes them much useful via a common metaphor. TensorFlow uses Python to give its users a front-end API that can be used for building applications, with the applications being executed in high-performance C++.

TensorFlow can be used for building, training, and running deep neural networks for image recognition, handwritten digit classification, recurrent neural networks, word embedding, natural language processing, etc.

Chapter 1: History of Machine Learning

The history of Machine Learning begins with the history of computing. And this history began before computers were even invented. The function of computers is actually math, in particular, Boolean logic. So before anyone could create a computing machine, the theory about how such a machine could work had to be figured out by mathematicians. It was the growth of this theory from hundreds of years ago that made possible the revolution in computer software that we see today. In a way, the present and future of computing and machine intelligence belong to great minds from our past.

In 1652, Blaise Pascal, a 19-year-old teenager, created an arithmetic machine that could add, subtract, multiply, and divide.

In 1679, German mathematician Gottfried Wilhelm Leibniz created a system of binary code that laid the groundwork for modern computing.

In 1834, English inventor Charles Babbage conceived a mechanical device that could be programmed with punch cards. While it was never actually built, its logical structure, Boolean logic, is what nearly every modern computer relies on to function.

In 1842, Ada Lovelace became the world's first computer programmer. At 27 years of age, she designed an algorithm for solving mathematical problems using Babbage's punch-card technology.

In 1847, George Boole created algebra capable of reducing all values to Boolean results. Boolean logic is what CPUs use to function and make decisions.

In 1936, Alan Turing discussed a theory describing how a machine could analyze and execute a series of instructions. His proof was published and is considered the base of modern computer science.

In 1943, a neurophysiologist, Warren McCulloch, and a mathematician, Walter Pitts, co-wrote a paper theorizing how neurons in the human brain might function. Then they modeled their theory by building a simple neural network with electrical circuits.

In 1950, Alan Turing proposed the notion of a “learning machine.” This machine could learn from the world like human beings do and eventually become artificially intelligent. He theorized about intelligence and described what was to become known as the “Turing Test” of machine intelligence. Intelligence, Turing mused, wasn’t well defined, but we humans seem to have it and to recognize it in others when we experience them using it. Thus, should we encounter a computer and can’t tell it is a computer when we interact with it, then we could consider it intelligent too.

In 1951, Marvin Minsky, with the help of Dean Edmonds, created the first artificial neural network. It was called SNARC (Stochastic Neural Analog Reinforcement Calculator).

In 1952, Arthur Samuel began working on some of the first machine learning programs at IBM's Poughkeepsie Laboratory. Samuel was a pioneer in the fields of artificial intelligence and computer gaming. One of the first things they built was a machine that could play checkers. More importantly, this checkers-playing program could learn and improve its game.

Machine Learning was developed as part of the quest in the development of Artificial Intelligence. The goal of Machine Learning was to have machines learn from data. But despite its early start, Machine Learning was largely abandoned in the development of Artificial Intelligence. Like work on the perceptron, progress in Machine Learning lagged as Artificial Intelligence moved to the study of expert systems.

Eventually, this focus on a logical, knowledge-based approach to Artificial Intelligence caused a split between the disciplines. Machine Learning systems suffered from practical and theoretical problems in representation and acquiring large data sets to work with. Expert systems came to dominate by 1980, while statistical and probabilistic systems like Machine Learning fell out of favor. Early neural network research was also abandoned by Artificial Intelligence researchers and became its own field of study.

Machine Learning became its own discipline, mostly considered outside the purview of Artificial Intelligence, until the 1990s. Practically, all of the progress in Machine Learning from the 1960s through to the 1990s was theoretical, mostly statistics and probability theory. But while not much

seemed to be accomplished, the theory and algorithms produced in these decades would prove to be the tools needed to re-energize the discipline. At this point, in the 1990s, the twin engines of vastly increased computer processing power and the availability of large datasets brought on a sort of renaissance for Machine Learning. Its goals shifted from the general notion of achieving artificial intelligence to a more focused goal of solving real-world problems, employing methods it would borrow from probability theory and statistics, ideas generated over the previous few decades. This shift and the subsequent successes it enjoyed brought the field of Machine Learning back into the fold of Artificial Intelligence, where it resides today as a sub-discipline under the Artificial Intelligence umbrella.

However, Machine Learning was, continues to be, and might remain a form of Specific Artificial Intelligence. SAI are software algorithms able to learn a single or small range of items, which cannot be generalized to the world at large. To this date, GAI remains an elusive goal, one many believe will never be reached. Machine Learning algorithms, even if they are specific artificial intelligence (SAI), are changing the world and will have an enormous effect on the future.

Chapter 2: Theories of Machine Learning

The goal of Machine Learning stated simply is to create machines capable of learning about the world so they can accomplish whatever tasks we want them to do. This sounds simple enough. This is something every human being accomplishes without any effort. From birth until adulthood, human beings learn about the world until they are able to master it. Why should this goal be so difficult for machines?

But even after 70 years of dedicated effort, the goal of general artificial intelligence remains elusive. As of today, there is nothing even approaching what could be called a Generalized Artificial Intelligence (GAI). That is an AI capable of learning from experience with the world, and from that learning becoming capable of acting in the world with intelligence. Like a child transitions from a relatively helpless being at birth, spends the next decade or two experiencing its world, language, culture, and physical reality surrounding it, then finally becomes an intelligent adult with an astounding general intelligence. A human being can learn to solve crossword puzzles in seconds, merely by a quick examination of the wording of the questions and the design of the field of squares. Drawing on knowledge of language, puzzles, and the fact we often put letters in little boxes, a human being can infer how to do a crossword from previous, unrelated experiences. This simple task would absolutely baffle the most powerful artificial intelligence on Earth if it had not been specifically trained to do crossword puzzles.

Early Artificial Intelligence started out as an attempt to understand and employ the rules of thought. The assumption was that thinking is a chain of logical inference. Such a rules-based system might work like this:

Assertion: Birds can fly.

Fact: A robin is a bird.

Inference: Robins can fly.

Therefore, early Artificial Researchers claimed, encoding thousands of such inferences in massive databases would allow a machine to make intelligent

claims about the world. A machine equipped with such a database should be able to make inferences about the world and display intelligence to rival human intelligence.

Unfortunately, researchers using this rules-based approach very quickly ran into problems. To continue with our bird example, what about flightless birds like the ostrich, or penguin, or emu? What about a bird with a broken wing that, at least temporarily, cannot fly? What do we call a bird in a small cage, where it cannot fly because of the bars? To human minds confronted with these exceptions, these questions are simple — of course, they are birds because flight, although available to most birds, does not define bird-ness.

Rules-based Artificial Intelligence researchers developed language to accommodate their rules of inference, and these languages were flexible enough to accept and deal with such deviations as flightless birds. But writing down and programming all these distinctions, the deviations from the rule proved far more complicated than anticipated. Imagine attempting to code every exception to every rule of inference. The task quickly becomes Herculean and impossible for mere mortals.

The rise of probabilistic systems was in response to this choke point for these inference methods of artificial intelligence. Probabilistic systems are fed enormous amounts of relevant data about a subject like birds and are left to infer what they have in common (sometimes guided with classifying labels — supervised or semi-supervised, sometimes with no labels at all — unsupervised). These commonalities are used to modify the algorithm itself until its output approaches the desired output for the data in question.

This system works okay for birds or cats, but it does not work well for a more abstract idea like flight. Being fed images of thousands of things flying might allow an Artificial Intelligence to understand the concept of flight, but this learning tends to also label clouds as flying, or the sun, or a picture of an antenna on the top of a building with the sky as a backdrop. And the concept of flight is much more concrete than something truly abstract, like love or syntax.

So probabilistic Artificial Intelligence, what we are calling Machine Learning, switched from attempting to mimic or match human thought and to

deal with concrete, real-world problems that it could solve. This transition in goals sparked the Machine Learning revolution. By focusing on concrete, real-world problems rather than generalized ideas, the algorithms of Machine Learning were able to achieve very powerful, very precise predictions of future states and to apply those predictions to new data.

Machine Learning today is considered a subset of Artificial Intelligence. The purpose of Machine Learning is to develop software that can learn for itself. Essentially, researchers present a set of data with labels to a piece of software. The software ingests these examples and the relevant labels provided and attempt to extrapolate rules from the data and labels in order to make decisions on data it has never seen before. This rules-based system to make new matches is called a classifier.

The name for this approach to handling data is Computational Learning Theory. It is highly theoretical and mathematical, using statistics in particular. Computation Learning Theory is the study and analysis of machine learning algorithms. Learning Theory deals with inductive learning, a method known as supervised learning.

The goal of supervised learning is to correctly label new data as it is encountered, in the process reducing both the number of mislabels and the speed of the pattern matching process itself. However, there are hard limits to the performance of Machine Learning algorithms. The facts are data sets used for training are always finite, and the future is uncertain. These facts mean that the results of Machine Learning systems will be probabilistic — it is very difficult to guarantee a particular algorithm's accuracy. Instead of guarantees, researchers will place upper and lower bounds on the probability of an algorithm's success at classification.

Chapter 3: Approaches to Machine Learning

There is no formal definition of the approaches to Machine Learning. However, it is possible to separate the approaches into five loose groups. This chapter will identify these groups and describe their individual philosophical approaches to Machine Learning. Each of these philosophies has been successful in solving different categories of problems that humans need to deal with.

Finally, we will examine three real-world methods employed in actual Machine Learning, where some of these philosophies are turned into working systems.

Philosophies of Machine Learning

Artificial Intelligence has always been inspired by human cognition. And even though the philosophical methods described below rely heavily on complex math and statistics, each of them is an attempt to mimic, if not the process of human cognition, then at least the means whereby human cognition seems to function. This is an artificial intelligence that does not concern itself with how people or machines think, but only with whether the results of such thinking produce meaningful results.

Inverse Deduction

Human beings are hard-wired to fill in gaps. From the attachment of the ocular nerve to the retina (creating a black spot in our vision where we can't actually see) to creating faces where none exist (the face in the moon, for example), our brains are designed to fill in details in order to make patterns. This is a form of pattern induction. Deduction is the opposite. It is the process of moving from general experience to individual details. Inductive Artificial Intelligence is a more mathematical approach to this gap in the pattern filling. For example, if our algorithm knows $5 + 5 = 10$, then it still is able to induce an answer if we give it 5 and ask it what needs to be added to this 5 to end up with 10. The system must answer the question "what knowledge is missing?" The required knowledge to answer this question is generated through the ingestion of data sets.

Making Connections

Another philosophy of Artificial Intelligence attempts to mimic some functions of the human brain. Scientists have been studying how human brains function for decades, and some researchers in Artificial Intelligence, beginning in earnest in the 1950s, have attempted to represent, mathematically, the neural architecture of the human brain. We call this approach the neural network. They are composed of many mathematical nodes, analogous to neurons in the brain. Each node has an input, a weight, and an output. They are often layered, with the output of some neurons becoming the input of others. These systems are trained on data sets, and with

each iteration, the weights attributed to each node is adjusted as required in order to more closely approximate the desired output. The neural network approach has had much of the success in machine learning, lately in particular, because it is suited for dealing with large amounts of data. It is important to understand, however, that the term neural net is more of an inspiration than an attempt to replicate how human brains actually function. The complex interconnected web of neurons in the human brain is not something a neural net attempts to emulate. Instead, it is a series of mathematical, probabilistic inputs and outputs that attempts to model the results of the human neural activity.

Evolution

This approach is to mimic biological evolution. A variety of algorithms are tasked with solving a problem and exposed to an environment where they compete to produce the desired output. Each individual algorithm has a unique set of characteristics with which it can attempt to accomplish the task. After each iteration, the most successful algorithms are rewarded just like a successful animal in nature with higher fitness, meaning they are more likely to pass their features on to the next generation. Those that do continue to the next iteration can cross their feature sets with other successful algorithms, mimicking sexual reproduction. Add in some random “mutations,” and you have the next generation, ready to iterate over the data again. This method can produce some powerful learning in its algorithms.

Bayesian

The Bayesian approach uses probabilistic inference to deal with uncertainty. It is a method of statistical inference where Bayes’ theorem is employed to update hypothesis probabilities as new evidence becomes available. After a number of iterations, some hypotheses become more likely than others. A real-world example of this approach is Bayesian spam filtering. How it works is fairly simple. It has a collection of words that it compares to in the content of emails. It compares these words to the contents of email messages, increasing or decreasing the probability of the email being spam according to what it finds. Bayesian algorithms have been quite successful over the last two decades because they are adept at building structured models to real-world problems which have previously been considered intractable.

By Analogy

The fifth philosophical approach to Machine Learning involves using an analogy. This is yet another tactic borrowed from human cognition. Reasoning by analogy is a powerful tool humans use to incorporate new information by comparing it to similar information that is already known. By comparing this new information to established categories in our mental repertoire, human beings are able to classify and incorporate new information even when very little detail about this new information has been provided. The label for this approach is the “nearest neighbor” principle – basically asking what this new information is more similar to and categorizing it based on its similarity to known items. This approach is used in Support Vector Machines and until recently was likely the most powerful Machine Learning philosophy in use. An example of a real-world Support Vector Machine can be found in features like movie recommendations. If you and a stranger have given 5 stars to one movie and only one star to a different movie, the algorithm will recommend another movie that this stranger has rated favorably, assuming by analogy that your tastes are similar, and therefore, you will appreciate this movie recommendation as well.

What these philosophical approaches to Machine Learning tell us is twofold — the discipline, despite its roots in the mid-20th century, is still a nascent discipline that continues to grow into its own understanding. After all, Machine Learning is attempting to model the capacity of the most complex device in the known universe, the human brain.

As the history of artificial intelligence demonstrates, we can see there is no “best” approach to Machine Learning. The “best” approach is only the best approach for now. Tomorrow, a new insight might surpass everything that’s been accomplished to date, and pave a new path for artificial intelligence to follow.

How do collections of computer code and mathematics thus far managed to learn? Here are the three most common methods currently in use to get computer algorithms to learn.

Supervised and Semi-supervised Learning Algorithms

In the supervised approach to Machine Learning, researchers first construct a mathematical model of a data set which includes the expected inputs and the required outputs. The data this produces is called training data and consists of sets of training examples (input and desired output). Each of these training examples is comprised of one or more inputs, along with the desired output – the supervisory signal. In semi-supervised learning, some of the desired output values are missing from the training examples.

Through iteration, that is, running the training data through the learning algorithm repeatedly, the learning algorithm develops a function to match the desired outputs from the inputs of the training data. During the iteration, this function is optimized by the learning algorithm. When this system is deemed ready, new data sets are introduced that are missing the desired outputs – these are known as the testing sets. At this point, errors or omission in the training data may become more obvious, and the process can be repeated with new or more accurate output requirements. An algorithm that successfully modifies itself to improve the accuracy of its predictions or outputs can be said to have successfully learned to perform a task.

Supervised and semi-supervised learning algorithms include two classes of data handling — classification and regression. When outputs are limited by a constrained set of possible values, classification learning algorithms are employed. But when the outputs can be returned as a numerical value in a range of possible values, regression learning algorithms are the best fit.

Finally, there is similarity learning. While closely related to regression and classification learning, the goal of similarity learning is to examine data sets and determine how similar or different the information sets are. Similarity learning can be used in tasks such as ranking, recommendation systems (Netflix recommended anyone?), visual identity tracking, facial recognition/verification, and speaker (audio) verification.

Unsupervised Learning Algorithms

In unsupervised learning, algorithms are simply given large data sets containing only inputs. Their goal is to find grouped or clustered data points that can be compared to new data sets. Unsupervised learning algorithms learn from data that has no labels and has not been organized or classified before being submitted. Instead of attempting to produce a required output prompted by supervisory systems, these algorithms attempt to find commonalities in the inputs they receive, which they then apply to new data sets. They react when these commonalities are found, missing, or broken in each new data set. Unsupervised learning algorithms are used in diverse fields including density estimation in statistics and the summarizing and explanation of data features. This type of learning can also be useful in fraud detection, where the goal is to find anomalies in input data.

Cluster analysis is when unsupervised learning algorithms break down a set of observations about data in clusters (subsets) so that the information within each cluster is similarly based on one or more predefined criteria. Information drawn from other clusters will be internally similar, while dissimilar from each other. There are different approaches to data clustering, which are derived from making alternative assumptions about the structure of the data.

Reinforcement Learning

Reinforcement learning is an approach to Machine Learning that attempts to “reward” systems for taking actions in their environment that are in alignment with the objectives of the system. In this way, software trains itself using trial and error over sets of data until it induces the reward state. This field is actually quite large and does not solely come the purview of Machine Learning. It is also employed in disciplines like game theory, swarm intelligence, simulation-based optimization, information theory, and more.

Reinforcement learning for Machine Learning often employs an environment characterized by a Markov Decision Process – that is, a mathematical model to deal with a situation where the cause of the outcome is part random and part under control of the software decision maker. In addition, reinforcement learning algorithms typically employ dynamic programming techniques. This is a method whereby tasks are broken down into sub-tasks, where possible, to be regrouped once each sub-task has been accomplished into a solution for the main task. Reinforcement learning algorithms do not need to assume there is an exact model of the Markov Decision Process for them to employ. Instead, these algorithms are successfully employed when such exact models are not feasible.

Typical uses for Machine Learning reinforcement learning algorithms are in learning to play games against human opponents or in self-driving/autonomous vehicles.

Chapter 4: Environment Setup

Before getting into the practical part of machine learning and deep learning, we need to install our two libraries, that is, Scikit-Learn and TensorFlow.

Installing Scikit-Learn

Scikit-learn is supported in Python 2.7 and above. Before installing Scikit-learn, ensure that you have Numpy and SciPy libraries already installed. Once you have installed these, you can go ahead to install Scikit-learn on your machine.

The installation of these libraries can be done using pip. Pip is a tool that comes with Python, which means that you get pip after installing Python. To install sci-kit-learn, run the following command on the terminal of your operating system:

```
pip install sci-kit-learn
```

The installation should run and come to completion.

You can also use conda to install scikit-learn.

```
conda install scikit-learn
```

Once the installation of scikit-learn is complete, you need to import it into your Python program in order to use its algorithms. This can be done by using the **import** statement as shown below:

```
import sklearn
```

If the command runs without an error, know that the installation of scikit-learn was successful. If the command generates an error, note that the installation was not successful.

You can now use scikit-learn to create your own machine learning models.

Installing TensorFlow

TensorFlow comes with APIs for programming languages like C++, Haskell, Go, Java, Rust, and it comes with a third-party package for R known as **tensorflow**. On Windows, TensorFlow can be installed with pip or Anaconda.

The native pip will install the TensorFlow on your system without having to go through a virtual environment. However, note that the installation of TensorFlow with pip may interfere with other Python installations on your system. However, the good thing is that you only have to run a single command and TensorFlow will be installed on your system. Also, when TensorFlow is installed via pip, users will be allowed to run the TensorFlow programs from the directory they want.

To install TensorFlow with Anaconda, you may have to create a virtual environment. However, within the Anaconda itself, it is recommended that you install TensorFlow via the **pip install** command rather than the **conda install** command.

Ensure that you have installed Python 3.5 and above on your Windows system. Python3 comes with a pip3 program which can be used for installation of TensorFlow. This means we should use the **pip3 install** command for installation purposes. The following command will help you install the CPU-only version for TensorFlow:

```
pip3 install --upgrade tensorflow
```

The command should be run from the command line:

Figure 1: Information obtained from Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

If you need to install a GPU version for TensorFlow, run this command:

```
pip3 install --upgrade tensorflow-gpu
```

This will install TensorFlow on your Windows system.

You can also install TensorFlow with the Anaconda package. Pip comes installed with Python, but Anaconda doesn't. This means that to install TensorFlow with Anaconda, you should first install the Anaconda. You can download Anaconda from its website and find the installation instructions from the same site.

Once you install Anaconda, you get a package named **conda**, which is good for the management of virtual environments and the installation of packages. To get to use this package, you should start the Anaconda.

On Windows, click Start, choose **“All Programs,”** expand the **“Anaconda ...”** folder then click the **“Anaconda Prompt.”** This should launch the anaconda prompt on your system. If you need to see the details of the conda package, just run the following command on the terminal you have just opened:

```
conda info
```

This should return more details regarding the package manager.

Figure 2: Information from Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

There is something unique with Anaconda. It helps us create a virtual Python environment using the conda package. This virtual environment is simply an isolated copy of Python with the capability of maintaining its own files, paths, and directories so that you may be able to work with specific versions of Python or other libraries without affecting your other Python projects (Samuel, 2018). Virtual environments provide us with a way of isolating projects and avoid problems that may arise as a result of version requirements and different dependencies across various components. Note that this virtual environment will remain separate from your normal Python environment meaning that the packages installed in the virtual environment will not affect the ones you have in your Python's normal environment.

We need to create a virtual environment for the TensorFlow package. This can be done via the **conda create** command. The command takes the syntax given below:

```
conda create -n [environment-name]
```

In our case, we need to give this environment the name **tsorenvirion**. We can create it by running the following command:

```
conda create -n tsorenvirion
```

Allow the process of creating the environment to continue or not. Type “y” and hit the enter key on the keyboard. The installation will continue successfully.

Figure 3: Information obtained from Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

After creating an environment, we should activate it so that we may be able to use it. The activation can be done using the **activate** command followed by the name of the environment as shown below:

```
activate tsorenvirion
```

Now that you have activated the TensorFlow environment, you can go ahead and install TensorFlow package in it. You can achieve this by running the following command (Newell, 2019):

```
conda install tensorflow
```

You will be presented with a list of packages that will be installed together with the TensorFlow package. Just type “y,” then hit the enter key on your keyboard. The installation of the packages will begin immediately. The process may take a number of minutes, so remain patient. However, the speed of your internet connection will determine how long this process takes. The progress of the installation process will also be shown on the prompt

window.

Figure 4: Information obtained from Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

```

(tensorflowenv) C:\Users\admin\Documents>conda install tensorflow
Fetching package metadata .....
Solving package specifications: .

Package plan for installation in environment C:\Users\admin\Anaconda3\envs\tensorflowenv:

The following NEW packages will be INSTALLED:

 _tfflow_select:      2.1.0-gpu
 absl-py:             0.6.1-py36_0
 astor:               0.7.1-py36_0
 blas:                1.0-mkl
 certifi:             2018.10.15-py36_0
 cudatoolkit:         9.0-1
 cudnn:               7.1.4-cuda9.0_0
 gast:                0.2.0-py36_0
 grpcio:              1.12.1-py36h1a1b453_0
 h5py:                 2.8.0-py36h3bdd7fb_2
 hdf5:                1.10.2-hac2f561_1
 icc_rt:              2017.0.4-h97af966_0
 intel-openmp:         2019.0-118
 keras-applications:  1.0.6-py36_0
 keras-preprocessing: 1.0.5-py36_0
 libprotobuf:         3.6.1-h7bd577a_0
 markdown:            3.0.1-py36_0
 mkl:                 2019.0-118
 mkl_fft:             1.0.6-py36hdbbbee80_0
 mkl_random:          1.0.1-py36h77b88f5_1
 numpy:               1.15.3-py36ha559c80_0
 numpy-base:          1.15.3-py36h8128ebf_0
 pip:                 10.0.1-py36_0
 protobuf:            3.6.1-py36h33f27b4_0
 python:              3.6.7-h33f27b4_1
 scipy:               1.1.0-py36h4f6bf74_1
 setuptools:          40.5.0-py36_0
 six:                 1.11.0-py36_1
 tensorboard:         1.11.0-py36he025d50_0
 tensorflow:           1.11.0-gpu_py36h5dc63e2_0
 tensorflow-base:     1.11.0-gpu_py36h6e53903_0
 termcolor:           1.1.0-py36_1
 vc:                  14.1-h0510ff6_4
 vs2015_runtime:      14.15.26706-h3a45250_0
 werkzeug:            0.14.1-py36_0
 wheel:               0.32.2-py36_0
 wincertstore:         0.2-py36h7fe50ca_0
 zlib:                1.2.11-h8395fce_2

Proceed [y/n]? y

 _tfflow_select- 100% !#####! Time: 0:00:00 598.96 kB/s
 blas-1.0-mkl.t 100% !#####! Time: 0:00:00 475.58 kB/s
 cudatoolkit-9. 100% !#####! Time: 0:21:50 271.86 kB/s
 intel-openmp-2 100% !#####! Time: 0:00:06 274.72 kB/s
 vs2015_runtime 100% !#####! Time: 0:00:08 275.83 kB/s
 cudnn-7.1.4-cu 100% !#####! Time: 0:12:12 275.45 kB/s

```

After some time, the installation process will complete, and it will be time for you to verify whether the installation was successful. We can simply do this by running the Python's **import** statement. The statement should be run from

Python's terminal. While on the Anaconda prompt, type **python** and hit the enter key. This should take you to the Python terminal. Now run the following **import** statement:

```
import tensorflow as tf
```

If the package was not installed successfully, you would get an error; otherwise, the installation of the package was successful.

Chapter 5: Using Scikit-Learn

Now that you are done with the installations, you can begin to use the libraries. We will begin with the Scikit-Learn library.

To be able to use scikit-learn in your code, you should first import it by running this statement:

```
import sklearn
```

Loading Datasets

Machine learning is all about analyzing sets of data. Before this, we should first load the dataset into our workspace. The library comes loaded with a number of datasets that we can load and work with. We will demonstrate this by using a dataset known as **Iris**. This is a dataset of flowers.

Regression

Linear regression

It is the most popular type of predictive analysis. Linear regression involves the following two things:

1. Do the predictor variables forecast the results of an outcome variable accurately?
2. Which particular variables are key predictors of the final variable, and in what standard does it impact the outcome variable?

Naming variables

The regression's dependent variable has many different names. Some names include outcome variable, criterion variable, and many others. The independent variable can be called exogenous variable or repressors.

Functions of the regression analysis

1. Trend Forecasting
2. Determine the strength of predictors
3. Predict an effect

Breaking down regression

There are two basic states of regression-linear regression and multiple regression. Although there are different methods for complex data and analysis, linear regression contains an independent variable to help forecast the outcome of a dependent variable. On the other hand, multiple regression has two or more independent variables to assist in predicting a result.

Regression is very useful to financial and investment institutions because it is used to predict the sales of a particular product or company based on the previous sales and GDP growth among many other factors. The capital pricing model is one of the most common regression models applied in finance. The example below describes the formulae used in linear and

multiple regression.

Figure 5: Information obtained from Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

```
Linear Regression:  $Y = a + bX + u$   
Multiple Regression:  $Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_tX_t + u$   
In this case:  
Y = variable which you want to predict (dependent variable)  
X = variable which you are using to predict Y (independent variable)  
a = the intercept  
b = the slope  
u = regression residual
```

Choosing the best regression model

Selecting the right linear regression model can be very hard and confusing. Trying to model it with a sample data cannot make it easier. This section reviews some of the most popular statistical methods which one can use to choose models and challenges that you might come across. It also lists some practical advice to use to select the correct regression model.

It always begins with a researcher who would like to expand the relationship between the response variable and predictors. The research team that is accorded with the responsibility to perform investigation essentially measures a lot of variables but only has a few in the model. The analysts will make efforts to reduce the variables that are different and apply the ones which have an accurate relationship. As time moves on, the analysts continue to add more models.

Statistical methods to use to find the best regression model

If you want a great model in regression, then it is important to take into consideration the type of variables which you want to test as well as other variables which can affect the response.

Modified R-squared and Predicted R-squared.

Your model should have higher modified and predicted R-squared values. The statistics are shown below help eliminate critical issues which revolve around R-squared.

- The adjusted R-squared increases once a new term improves the model.
- Predicted R-squared belongs to the cross-validation that helps define the manner in which your model can generalize remaining data sets.

P-values for the Predictors

When it comes to regression, a low value of P denotes statistically significant terms. The term “Reducing the model” refers to the process of factoring in all candidate predictors contained in a model.

Stepwise regression

This is an automated technique which can select important predictors found in the exploratory stages of creating a model.

Real World Challenges

There are different statistical approaches for choosing the best model. However, complications still exist.

- The best model happens when the variables are measured by the study.
- The sample data could be unusual because of the type of data collection method. A false positive and false negative process happens when you handle samples.
- If you deal with enough models, you’ll get variables that are significant but only correlated by chance.
- P-values can be different depending on the specific terms found in the model.
- Studies have discovered that the best subset regression and stepwise regression can’t select the correct model.

Finding the correct Regression Model

Theory

Study research done by other experts and reference it in your model. It is important that before you start regression analysis, you should develop ideas about the most significant variables. Developing something based on

outcome from other people eases the process of collecting data.

Complexity

You may think that complex problems need a complex model. Well, that is not the case because studies show that even a simple model can provide an accurate prediction. Once there is a model with the same explanatory potential, the simplest model is likely to be a perfect choice. You just need to start with a simple model as you slowly advance the complexity of the model.

How to calculate the accuracy of the predictive model

There are different ways in which you can compute the accuracy of your model. Some of these methods are listed below:

- 1. You divide the data sets into test and training data sets. Next, build the model based on the training set and apply the test set as a holdout sample to measure your trained model with the test data.*
- 2. Another method is to calculate the “Confusion Matrix” to the computer False Positive Rate and False Negative Rate. These measures will allow a person to choose whether to accept the model or not. If you consider the cost of the errors, it becomes a critical stage of your decision whether to reject or accept the model.*
- 3. Computing Receiver Operating Characteristic Curve (ROCC) or the Lift Chart or Area under the curve (AUC) are other methods that you can use to decide on whether to reject or accept a model.*

Chapter 6: k-Nearest Neighbors Algorithm

The KNN algorithm is highly used for building more complex classifiers. It is a simple algorithm, but it has outperformed many powerful classifiers. That is why it is used in numerous applications of data compression, economic forecasting, and genetics.

KNN is a supervised learning algorithm, which means that we are given a labeled dataset made up of training observations $(\mathbf{x}, \mathbf{y})$, and our goal is to determine the relationship between $\mathbf{x}$ and $\mathbf{y}$ (Karbhari, 2019). This means that we should find a function that $\mathbf{x}$ to $\mathbf{y}$ such that when we are given an input value for $\mathbf{x}$, we are able to predict the corresponding value for $\mathbf{y}$.

The concept behind the KNN algorithm is very simple. It calculates the distance of the new data point to all the other training data points. The distance can be of various types including Manhattan, Euclidean, etc. The K-nearest data points are chosen, in which K can be any integer. Finally, the data point is assigned to the class in which most of the K data points belong to.

We will use a dataset named Iris. We had explored it previously. We will be using this to demonstrate how to implement the KNN algorithm. This dataset is made up of four attributes namely **sepal-width**, **sepal-length**, **petal-width** and **petal-length**. Each type of the Iris plant has certain attributes. Our goal is to predict the class to which a plant belongs to. The dataset has three classes **Iris-setosa**, **Iris-versicolor**, and **Iris-virginica**.

We now need to load the dataset into our working environment. Download it from the following URL:

<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

Store it in a Pandas data frame. This following script will help us achieve this:

```
# create a variable for the dataset url
```

```
iris_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

`# Assign column names to the dataset`

```
iris_names = ['Slength', 'Swidth', 'Plength', 'Pwidth', 'Class']
```

`# Load the dataset from the url into a pandas dataframe`

```
dataset = pd.read_csv(iris_url, names=iris_names)
```

We can have a view of the first few rows of the dataset:

```
print(dataset.head())
```

	Slength	Swidth	Plength	Pwidth	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

The S is for Sepal while P is for Petal. For example, **Slength** represents **Sepal length** while **Plength** represents **Petal length**.

```
import sys
```

```
sys.__stdout__ = sys.stdout
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

`# Assign column names to our dataset`

```
names = ['Slength', 'Swidth', 'Plength', 'Pwidth', 'Class']
```

`# Read the dataset to a pandas dataframe`

```
dataset = pd.read_csv(url, names=names)
```

As usual, we should divide the dataset into attributes and labels:

```
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, 4].values
```

The variable X will hold the first four columns of the dataset which are the attributes while the variable y will hold the labels.

Splitting the Dataset

We need to be able to tell how well our algorithm performed. This will be done during the testing phase. This means that we should have training and testing data. We need to split the data into two such parts. 80% of the data will be used as the training set while 20% will be used as the test set.

Let us first import the **train_test_split** method from Scikit-Learn:

```
from sklearn.model_selection import train_test_split
```

We can then split the two as follows:

Feature Scaling

Before we can make the actual predictions, it is a good idea for us to scale the features. After that, all the features will be evaluated uniformly. Scikit-Learn comes with a class named **StandardScaler**, which can help us perform the feature scaling. Let us first import this class:

```
from sklearn.preprocessing import StandardScaler
```

We then instantiate the class, and use it to fit a model based on it:

```
feature_scaler = StandardScaler()
```

```
feature_scaler.fit(X_train)
```

```
X_train = feature_scaler.transform(X_train)
```

```
X_test = feature_scaler.transform(X_test)
```

The instance was given the name **feature_scaler**.

Training the Algorithm

With the Scikit-Learn library, it is easy for us to train the KNN algorithm. Let us first import the **KNeighborsClassifier** from the Scikit-Learn library:

```
from sklearn.neighbors import KNeighborsClassifier
```

The following code will help us train the algorithm:

```
knn_classifier = KNeighborsClassifier(n_neighbors=5)
```

```
knn_classifier.fit(X_train, y_train)
```

Note that we have created an instance of the class we have created and named the instance **knn_classifier**. We have used one parameter in the instantiation, that is, **n_neighbors**. We have used 5 as the value of this parameter, and this basically denotes the value of K. Note that there is no specific value for K, and it is chosen after testing and evaluation. However, for a start, 5 is used as the most popular value in most KNN applications.

We can then use the test data to make predictions. This can be done by running the script given below:

```
pred_y = knn_classifier.predict(X_test)
```

Evaluating the Accuracy of the Algorithm

Evaluation of the KNN algorithm is not done in the same way as evaluating the accuracy of the linear regression algorithm. We were using metrics like RMSE, MAE, etc. In this case, we will use metrics like confusion matrix, precision, recall, and f1 score.

We can use the **classification_report** and **confusion_matrix** methods to calculate these metrics. Let us first import these from the Scikit-Learn library:

```
from sklearn.metrics import confusion_matrix, classification_report
```

Run the following script:

```
print(confusion_matrix(y_test, pred_y))  
print(classification_report(y_test, pred_y))
```

```
[[13  0  0]  
 [ 0  5  0]  
 [ 0  3  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	13
Iris-versicolor	0.62	1.00	0.77	5
Iris-virginica	1.00	0.75	0.86	12
avg / total	0.94	0.90	0.90	30

The results given above show that the KNN algorithm did a good job in classifying the 30 records that we have in the test dataset. The results show that the average accuracy of the algorithm on the dataset was about 90%. This is not a bad percentage.

Comparing K Value with the Error Rate

We earlier said that there is no specific value of K that can be said to give the best results on the first go. We chose 5 because it is the most popular value used for K. The best way to find the best value of K is by plotting a graph of K value and the corresponding error for the dataset.

Let us create a plot using the mean error for predicted values of the test set for all the K values that range between 1 and 40. We should begin by calculating the mean of error for the predicted value with K ranging between 1 and 40. Just run the script given below:

```
error = []  
# K values range between 1 and 40  
for x in range(1, 40):  
    knn = KNeighborsClassifier(n_neighbors=x)  
    knn.fit(X_train, y_train)  
    pred_x = knn.predict(X_test)  
    error.append(np.mean(pred_x != y_test))
```

The code will run the loop from 1 to 40. In every iteration, the mean error for the predicted value of the test set is calculated and the result is appended to **error** list.

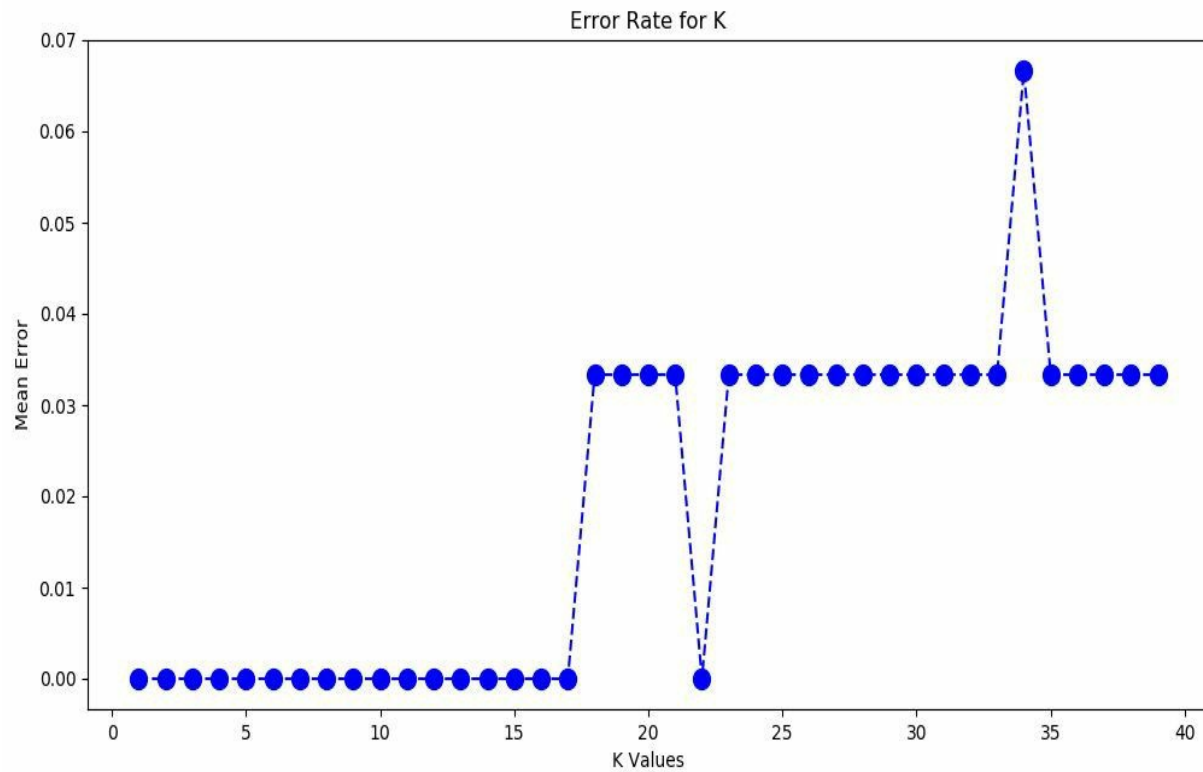
We should now plot the values of **error** against the values of K. The plot can be created by running the script given below:

```
plt.figure(figsize=(12, 6))  
plt.plot(range(1, 40), error, color='blue', linestyle='dashed', marker='o',  
         markerfacecolor='blue', markersize=10)  
plt.title('Error Rate for K')  
plt.xlabel('K Values')
```

```
plt.ylabel('Mean Error')
```

```
plt.show()
```

The code generates the plot given below:



The graph shows that we will get a Mean Error of 0 when we use values of K between 1 and 17. It will then be good for you to play around with the value of K and see its impact on the accuracy of the predictions.

Chapter 7: K-Means Clustering

Clustering falls under the category of unsupervised machine learning algorithms. It is often applied when the data is not labeled. The goal of the algorithm is to identify clusters or groups within the data.

The idea behind the clusters is that the objects contained in one cluster are more related to one another than the objects in the other clusters. The similarity is a metric reflecting the strength of the relationship between two data objects. Clustering is highly applied in exploratory data mining. It has many uses in diverse fields such as pattern recognition, machine learning, information retrieval, image analysis, data compression, bio-informatics, and computer graphics.

The algorithm forms clusters of data based on the similarity between data values. You are required to specify the value of K, which is the number of clusters that you expect the algorithm to make from the data. The algorithm first selects a centroid value for every cluster. After that, it performs three steps in an iterative manner:

1. Calculate the Euclidian distance between every data instance and the centroids for all clusters.
2. Assign the instances of data to the cluster of centroid with the nearest distance.
3. Calculate the new centroid values depending on the mean values of the coordinates of the data instances from the corresponding cluster.

Let us manually demonstrate how this algorithm works before implementing it on Scikit-Learn:

Suppose we have two dimensional data instances given below by the name `D`^[1]:

$$D = \{ (5,3), (10,15), (15,12), (24,10), (30,45), (85,70), (71,80), (60,78), (55,52), (80,91) \}$$

Our goal is to divide the data into two clusters, namely C1 and C2 depending

on the similarity between the data points.

We should first initialize the values for the centroids of both clusters, and this should be done randomly. The centroids will be named $c1$ and $c2$ for clusters $C1$ and $C2$ respectively, and we will initialize them with the values for the first two data points, that is, $(5,3)$ and $(10,15)$. It is after this that you should begin the iterations.

Anytime that you calculate the Euclidean distance, the data point should be assigned to the cluster with the shortest Euclidean distance. Let us take the example of the data point $(5,3)$:

Euclidean Distance from the Cluster Centroid $c1 = (5,3) = 0$

Euclidean Distance from the Cluster Centroid $c2 = (10,15) = 13$

The Euclidean distance for the data point from point centroid $c1$ is shorter compared to the distance of the same data point from centroid $c2$. This means that this data point will be assigned to the cluster $C1$.

Let us take another data point, $(15,12)$:

Euclidean Distance from the Cluster Centroid $c1 = (5,3)$ is 13.45

Euclidean Distance from the Cluster Centroid $c2 = (10,15)$ is 5.83

Centroid $c2$ is shorter. Thus, it will be assigned to the cluster $C2$.

Now that the data points have been assigned to the right clusters, the next step should involve calculations of the new centroid values. The values should be calculated by determining the means of the coordinates for the data points belonging to a certain cluster.

If, for example, for $C1$, we had allocated the following two data points to the cluster,

$(5, 3)$ and $(24, 10)$, the new value for x coordinate will be the mean of the two:

$$x = (5 + 24) / 2$$

$$x = 14.5$$

The new value for y will be:

$$y = (3 + 10) / 2$$

$$y = 13/2$$

$$y = 6.5$$

The new centroid value for the c1 will be **(14.5, 6.5)**.

This should be done for c2, and the entire process should be repeated. The iterations should be repeated until when the centroid values do not update anymore. This means if for example, you do three iterations, you may find that the updated values for centroids c1 and c2 in the fourth iterations are equal to what we had in iteration 3. This means that your data cannot be clustered any further.

You are now familiar with how the K-Means algorithm works. Let us discuss how you can implement it in the Scikit-Learn library.

Let us first import all the libraries that we need to use:

```
import matplotlib.pyplot as plt[2]
```

```
import numpy as np
```

```
from sklearn.cluster import KMeans
```

Data Preparation

We should now prepare the data that is to be used. We will be creating a **numpy array** with a total of 10 rows and 2 columns. So, why have we chosen to work with a numpy array? It is because Scikit-Learn library can work with the numpy array data inputs without the need for preprocessing. Let us create it:

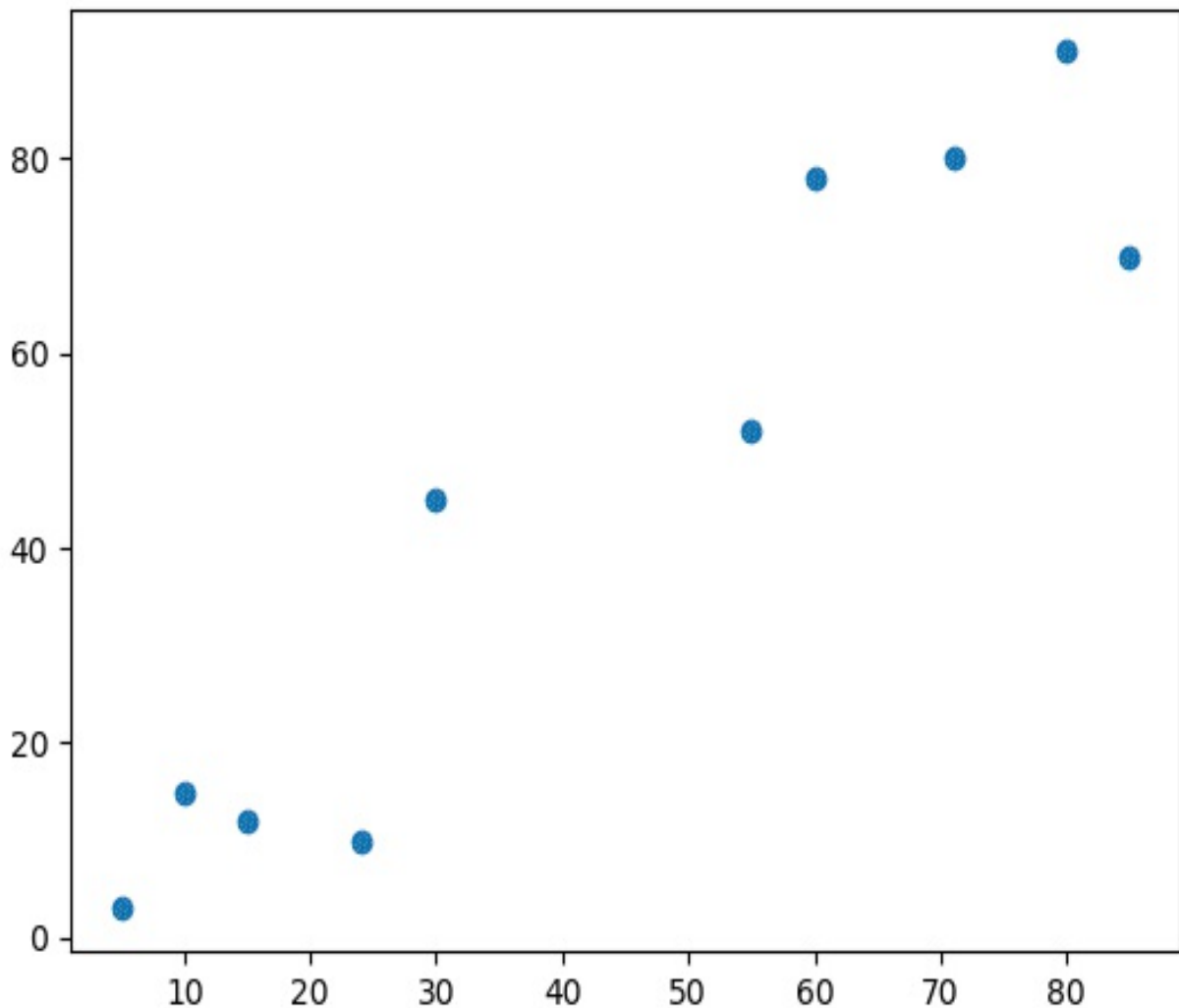
```
[80,91],])
```

Visualizing the Data

Now that we have the data, we can create a plot and see how the data points are distributed. We will then be able to tell whether there are any clusters at the moment:

plt.show()

The code gives the following plot:



If we use our eyes, we will probably make two clusters from the above data, one at the bottom with five points and another one at the top with five points.

We now need to investigate whether this is what the K-Means clustering algorithm will do.

Creating Clusters

We have seen that we can form two clusters from the data points; hence the value of K is now 2. These two clusters can be created by running the following code:

```
kmeans_clusters = KMeans(n_clusters=2)
kmeans_clusters.fit(X)
```

We have created an object named **kmeans_clusters**, and 2 have been used as the value for the parameter **n_clusters**. We have then called the **fit()** method on this object and passed the data we have in our numpy array as the parameter to the method.

We can now have a look at the centroid values that the algorithm has created for the final clusters:

```
print(kmeans_clusters.cluster_centers_)
```

This returns the following:

```
[[ 16.8  17. ]
 [ 70.2  74.2]]
```

The first row above gives us the coordinates for the first centroid, which is, **(16.8, 17)**. The second row gives us the coordinates of the second centroid, which is, **(70.2, 74.2)**. If you followed the manual process of calculating the values of these, they should be the same. This will be an indication that the K-Means algorithm worked well.

The following script will help us see the data point labels:

```
print(kmeans_clusters.labels_)
```

This returns the following:

```
[0 0 0 0 0 1 1 1 1 1]
```

The above output shows a one-dimensional array of 10 elements which correspond to the clusters that are assigned to the 10 data points. You clearly

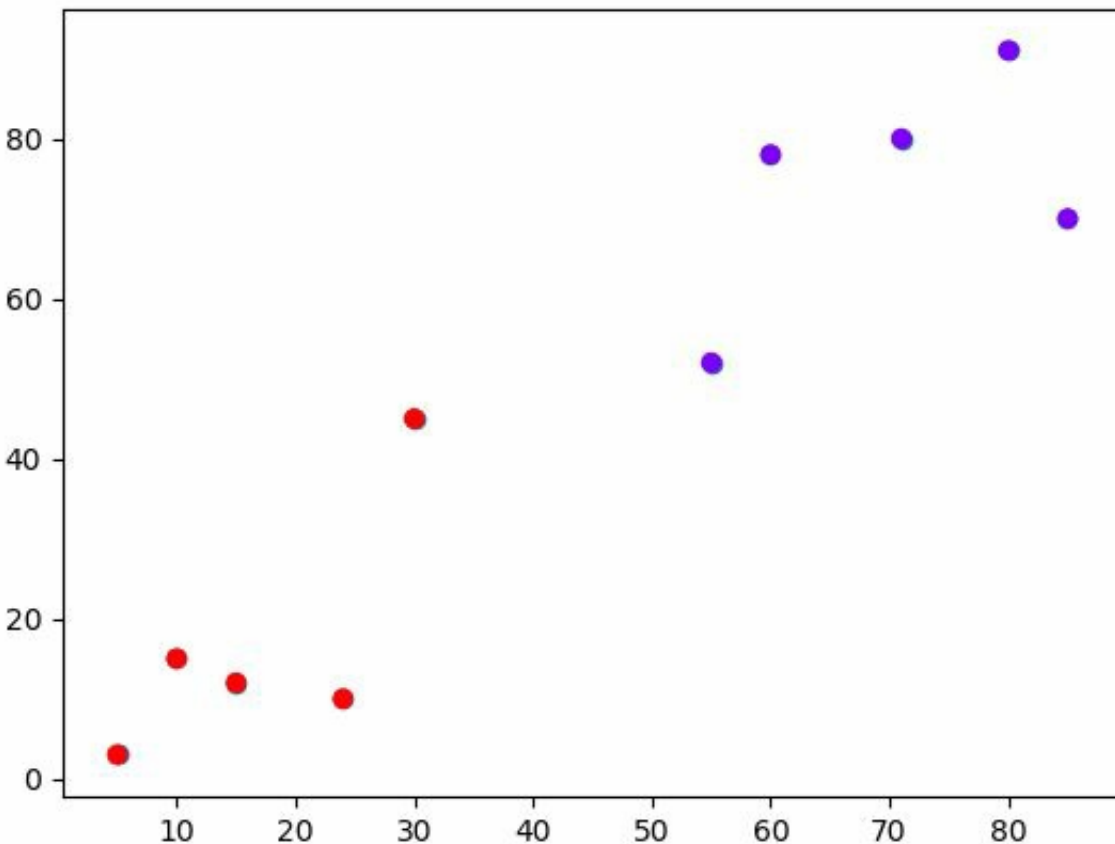
see that we first have a sequence of zeroes which shows that the first 5 points have been clustered together while the last five points have been clustered together. Note that the 0 and 1 have no mathematical significance, but they have simply been used to represent the cluster IDs. If we had three clusters, then the last one would have been represented using 2s.

We need to plot the data points alongside their assigned labels to be able to distinguish the clusters. Just execute the script given below:

```
plt.scatter(X[:,0],X[:,1], c=kmeans_clusters.labels_, cmap='rainbow')[3]
```

plt.show()

The script returns the following plot:



We have simply plotted the first column of the array named X against the second column. At the same time, we have passed **kmeans_labels** as the value for parameter c which corresponds to the labels. Note the use of the parameter **cmap='rainbow'**. This parameter helps us to choose the color type

for the different data points.

As you expected, the first five points have been clustered together at the bottom left and assigned a similar color. The remaining five points have been clustered together at the top right and assigned one unique color.

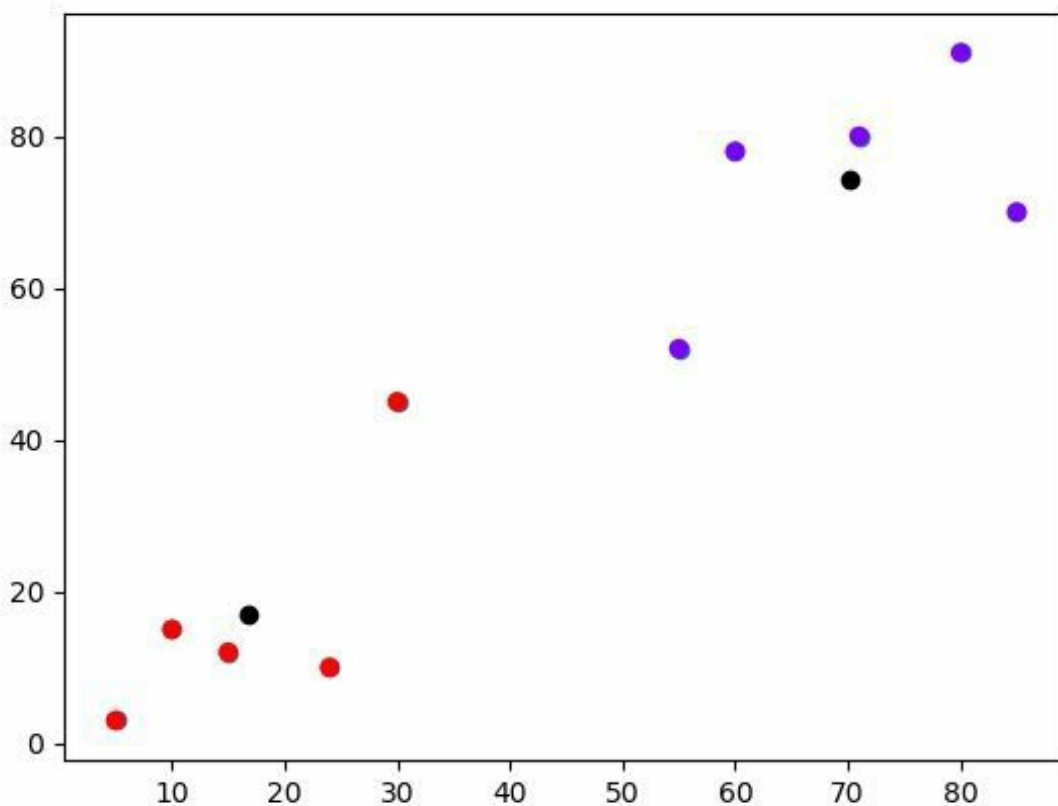
We can choose to plot the points together with the centroid coordinates for every cluster to see how the positioning of the centroid affects clustering. Let us use three clusters to see how they affect the centroids. The following script will help you to create the plot:

```
plt.scatter(X[:,0], X[:,1], c=kmeans_clusters.labels_, cmap='rainbow')[4]
```

```
plt.scatter(kmeans_clusters.cluster_centers_[:,0],  
kmeans_clusters.cluster_centers_[:,1], color='black')
```

plt.show()

The script returns the following plot:



We have chosen to plot the centroid points in black color.

Chapter 8: Support Vector Machines

SVMs fall under the category of supervised machine learning algorithms and are highly applied classification and regression problems. It is known for its ability to handle nonlinear input spaces. It is highly applied in applications like intrusion detection, face detection, classification of news articles, emails and web pages, handwriting recognition and classification of genes.

The algorithm works by segregating the data points in the best way possible. The distance between the nearest points is referred to as the **margin**. The goal is to choose a hyperplane with the maximum possible margin between the support vectors in a given dataset.

To best understand how this algorithm works, let us first implement it in Scikit-Learn library. Our goal is to predict whether a bank currency note is fake or authentic. We will use the attributes of the note including variance of the image, the skewness of the wavelet transformed image, the kurtosis of the image and entropy of the image. Since this is a binary classification algorithm, let us use the SVM classification algorithm.

If we have a linearly-separable data with two dimensions, the goal of a typical machine learning algorithm is to identify a boundary that will divide the data so as to minimize the misclassification error. In most cases, one gets several lines with all these lines correctly classifying the data.

SVM is different from the other classification algorithms in the way it selects the decision boundary maximizing the distance from the nearest data points for all classes. The goal of SVM is not to find the decision boundary only, but to find the most optimal decision boundary.

The most optimal decision boundary refers to the decision boundary with the maximum margin from the nearest points of all classes. The nearest points from the decision boundary maximizing the distance between the decision boundary and the points are known as **support vectors**. For the case of support vector machines, the decision boundary is known as **maximum margin classifier** or **maximum margin hyperplane**.

Complex mathematics is involved in the calculation of the support vectors. Determine the margin between the decision boundary and support vectors and maximize the margin.

Let us begin by importing the necessary libraries:

```
import numpy as np[5]
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

Importing the Dataset

We will use the **read_csv** method provided by the Pandas library to read the data and import it into our workspace (Agile Actors, 2018). This can be done as follows:

```
dataset = pd.read_csv("bank_note.csv")
```

Let us call the **shape** method to print the shape of the data for us:

```
print(dataset.shape)
```

This returns the following:

```
(1372, 5)
```

This shows that there are 1372 columns and 5 columns in the dataset. Let us print the first 5 rows of the dataset:

```
print(dataset.head())
```

Again, this may return an error because of the lack of the output information. Let us solve this using the Python's sys library. You should now have the following code:

```
import numpy as np[6]
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import sys
```

```
sys.__stdout__=sys.stdout
```

```
dataset = pd.read_csv("bank_note.csv")
```

```
print(dataset.head())
```

The code returns the following output:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

All attributes of the data are numeric as shown above. Even the last attribute is numeric as its values are either 0 or 1.

Preprocessing the Data

It is now time to subdivide the above data into attributes and labels as well as training and test sets. The following code will help us subdivide the data into attributes and labels:

```
X = dataset.drop('Class', axis=1)
```

```
y = dataset['Class']
```

The first line above helps us store all the columns of the dataset into variable **X**, except the **class** column. The **drop()** function has helped us exclude the **Class** column from this. The second line has then helped us store the **Class** column into variable **y**. The variable **X** now has attributes while the variable **y** now has the corresponding labels.

We have achieved the goal of dividing the dataset into attributes and labels. The next step is to divide the dataset into training and test sets. Scikit-learn has a library known as **model_selection** which provides us with a method named **train_test_split** that we can use to divide the data into training and test sets.

First, let us import the **train_test_split** method:

```
from sklearn.model_selection import train_test_split
```

Training the Algorithm

Now that the data has been split into training and test sets, we should now train the SVM on the training set. Scikit-Learn comes with a library known as **SVM** which has built-in classes for various SVM algorithms.

In this case, we will be doing a classification task. Hence, we will use the support vector classifier class (SVC). The takes a single parameter, that is, the kernel type. For a simple SVM, the parameter should be set to “**linear**” since the simple SVMs can only classify data that is linearly separable.

We will call the **fit** method of SVC to train the algorithm on our training set. The training set should be passed as a parameter to the **fit** method. Let us first import the SVC class from Scikit-Learn:

```
from sklearn.svm import SVC
```

Now run the following code:

```
svc_classifier = SVC(kernel='linear')
```

```
svc_classifier.fit(X_train, y_train)
```

Making Predictions

We should use the SVC class for making predictions. Note that the predictions will be made on the test data. Here is the code for making predictions:

```
pred_y = svc_classifier.predict(X_test)
```

Evaluating the Accuracy of the Algorithm

In classification tasks, we use confusion matrix, recall, precision, and F1 as the metrics. Scikit-Learn has the **metrics** library which provides us with the **confusion_matrix** and **classification_report** methods which can help us find the values of these metrics. The following code can help us find the value for these metrics:

First, let us import the above methods from the Scikit-Learn library:

```
from sklearn.metrics import confusion_matrix, classification_report
```

Here is the code that can help in doing the evaluation:

```
print(confusion_matrix(y_test,pred_y))
```

```
print(classification_report(y_test,pred_y))
```

The code returns the following:

```
[[160  1]
 [ 1 113]]
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	161
1	0.99	0.99	0.99	114
avg / total	0.99	0.99	0.99	275

The output given above shows that the algorithm did a good task; an average of 99% for the above metrics is not bad.

Let us give another example of how to implement SVM in Scikit-Learn using the Iris dataset. We had already loaded the Iris dataset, a dataset that shows details of flowers in terms of sepal and petal measurements, that is, width and length. We can now learn from the data, and then make a prediction for unknown data. These call for us to create an estimator, and then call its fit method.

This is demonstrated in the script given below:

```
from sklearn import svm
from sklearn import datasets
# Loading the dataset
iris = datasets.load_iris()
clf = svm.LinearSVC()
# learn from the dataset
clf.fit(iris.data, iris.target)
# predict unseen data
clf.predict([[ 6.2, 4.2, 3.5, 0.35]])
# Changing model parameters using the attributes ending with an underscore
print(clf.coef_ )
```

The code will return the following output:

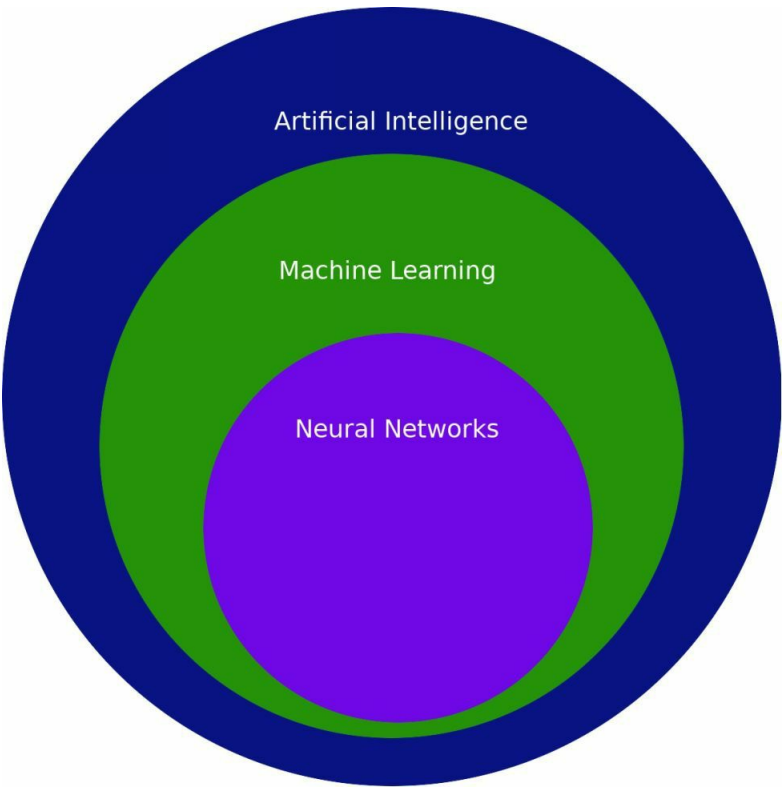
```
[[ 0.18423824  0.45123312 -0.80793878 -0.45071592]
 [ 0.05187834 -0.88969839  0.40345845 -0.93664852]
 [-0.85062306 -0.98667154  1.38105171  1.86536558]]
```

We now have the predicted values for our data. Note that we imported both **datasets** and **SVM** from the scikit-learn library. After loading the dataset, a model was fitted/created by learning patterns from the data. This was done by calling the **fit()** method. Note that the **LinearSVC()** method helps us to create an estimator for the support vector classifier, on which we are to create the model. We have then passed in new data for which we need to make a prediction.

Chapter 9: Machine Learning and Neural Networks

Neural networks were first developed in the late 1950s as a means to build learning systems that were modeled on our understanding of how the human brain functions. Despite their name, however, there is little resemblance between a neural network and a human brain. Mostly, the name serves as a metaphor and as inspiration. Each “neuron” in a Neural Network consists of a “node,” a piece of serial processing code designed to iterate over a problem until coming up with a solution at which point, the result is passed on to the next neuron in the layer, or to the next layer if the current layer’s processing is complete. In contrast, the human brain is capable of true parallel processing, by nature of the complex interconnections of its neurons and the fact its processing specialties are located in different areas of the brain (vision, hearing, smell, etc.), all of which can process signals simultaneously.

Neural networks are a sub-set of Machine Learning. They consist of software systems that mimic some aspects of human neurons. Neural Networks pass data through interconnected nodes. The data is analyzed, classified, and then passed on to the next node, where further classification and categorization may take place. The first layer of any Neural Network is the input layer, while the last is the output layer (these can be the same layer, as they were in the first, most primitive neural networks). Between these neural layers is any number of hidden layers that do the work of dealing with the data presented by the input layer. Classical Neural Networks usually contain two to three layers. When a Neural Network consists of more layers, it is referred to as Deep Learning. Such Deep Learning systems can have dozens or even hundreds of layers.



Feedforward Neural Networks

The first and simplest form of Neural Networks is called feedforward. As the name implies, data flows through a feedforward network in one direction, from the input, through the node layers containing the neurons and exits through the output layer. Unlike more modern neural networks, feedforward networks do not cycle or iterate over their data. They perform a single operation on the input data, and they provide their solution in an output stream.

Single-layer perceptron

This is the simplest form of a feedforward neural network with one layer of nodes. The input is sent to each node already weighted, and depending on how the node calculates the input and its weight, a minimal threshold may or not be met, and the neuron either fires (taking the “activated” value) or does not fire (taking the “deactivated” value).

Multi-layer perceptron

Multi-layer perceptrons, as the name suggests, consist of two or more (sometimes many more) layers, with the output of the upper layer becoming the input of the lower layer. Because there are many layers, this form of neural network often takes advantage of backpropagation, where the produced output is compared with the expected output, and the degree of error is fed back through the network to adjust the weights on the appropriate nodes, all with the intention of producing an output closer to the desired output state. Each error correction is tiny, so often a great number of iterations are required to achieve a “learned” state. At this point, the neural network is no longer considered a feedforward system proper. It is algorithms such as these multi-layer networks with backpropagation that have become some of the most successful and powerful Machine Learning devices in the field.

Recurrent Neural Networks

Recurrent Neural Networks propagate data in both directions, forward like feedforward networks but also backward from later to earlier processing stages. One of the main strengths of recurrent neural networks is the ability to remember previous states. Before recurrent neural networks, a neural network would forget previous tasks once it was asked to begin a new one. Imagine reading a comic but forgetting the content of the previous cells as you read the next one. Recurrent neural networks allow information to persist, meaning they can take input from subsequent events and “remember” their experience in previous ones. Put simply, recurrent neural networks are a series of input and output networks with the output of the first becoming the input of the second, the second’s output the input of the third, and so on. This cycling allows Recurrent Neural Networks to develop closer and closer approximations to the desired output.

Backpropagation

Theoretical development of backpropagation came from Paul Werbos’ Ph.D. thesis in 1974. Unfortunately, due to complications and difficulties encountered by those attempts to create Neural Networks (both theoretical and hardware-related), Neural Networks did not really catch on until the 1980s, when processor power and large, numerous data sets became available for training and testing.

The application of backpropagation is highly mathematical but can be summarized like this. Each iteration of a neural network produces a mathematical degree of error between the desired output and the actual output of the network. Each neuron in a neural network has a weight attached to it for the purposes of modifying its calculation on the input it receives. Backpropagation uses mathematics (probability statistics) when it is possible, to calculate the derivative of the error between expected and actual outputs. This derivative is then used on the next iteration to adjust the weight applied to each of the neurons. Each subsequent iteration produces a smaller error. Imagine a basketball dropped somewhere in a skateboard half-pipe. It will

roll down towards the center and up the other side, then reverse direction and roll towards the center again. Each time, it will roll less distance up the side before reversing direction. Eventually, gravity will bring the ball to a stable state in the center. In the same way, backpropagation reduces the error produced in each iteration as it brings the actual output closer to the desired output.

We have seen above that the goal of Machine Learning is to make software able to learn from the data it experiences. This is the goal of Neural Networks as well, but while Machine Learning makes decisions based on what data it has seen before, Neural Networks are designed to learn and make intelligent decisions on their own. This is particularly useful when the patterns being searched for are too numerous or complex for a software programmer to extract and submit as part of an input training data set.

Chapter 10: Machine Learning and Big Data

Big Data is pretty much what it sounds like — the practice of dealing with large volumes of data. And by large, we are talking about astoundingly huge amounts of data — gigabytes, terabytes, petabytes of data. A petabyte, to put this size into perspective, is 10^{15} bytes. Written out that is 1 PB = 1,000,000,000,000,000 bytes. When you consider that a single byte is equivalent in storage to a single letter like an ‘a’ or ‘x,’ the scale of the data sets being dealt with my Big Data is truly awe-inspiring. And these sizes are increasing every day.

The term Big Data comes from the 1990s, although computer scientists have been dealing with large volumes of data for decades. What sets Big Data apart from data sets before is the fact the size of data sets began to overwhelm the ability of traditional data analytics software to deal with it. New database storage systems had to be created (Hadoop for example) just to hold the data and new software written to be able to deal with so much information in a meaningful way.

Today the term Big Data brings with it a series of assumptions and practices that have made it a field all its own. Most Big Data discussions begin with the 3 V’s. Big data is data containing more variety arriving in increasing volumes and with increasing velocity (acceleration would be an accurate term to use here, but then we’d lose the alliteration).

Volume

The term volume refers to the vast amount of data available. When the term Big Data was coined in the early 2000s, the amount of data available for analysis was overwhelming. Since then, the volume of data created has grown exponentially. In fact, the volume of data produced has become so vast that new storage solutions have to be created just to deal with it. This increase in available data shows no sign of slowing, and data is, in fact, increasing geometrically by doubling every two years.

Velocity

Along with the rise in the amount of data being created, there has been an increase in the speed at which it is produced. Things like smartphones, RFID chips, and real-time facial recognition produce not only enormous amounts of data; this data is produced in real time and must be dealt with as it is created. If not processed in real time, it must be stored for later processing. The increasing speed of this data arriving strains the capacity of bandwidth, processing power, and storage space to contain it for later use.

Variety

Data does not get produced in a single format. It is stored numerically in detailed databases, produced in structure-less text and email documents, and stored digitally in streaming audio and video. There is stock market data, financial transactions, and so on, all of it uniquely structured. So not only must large amounts of data be handled very quickly, it is produced in many formats that require distinct methods of handling for each type.

Lately, two more V's have been added:

Value

Data is intrinsically valuable, but only if you are able to extract this value from it. Also, the state of input data, whether it is nicely structured in a numeric database or unstructured text message chains, affects its value. The less structure a data set has, the more work needs to be put into it before it can be processed. In this sense, well-structured data is more valuable than less-structured data.

Veracity

Not all captured data is of equal quality. When dealing with assumptions and predictions parsed out of large data sets, knowing the veracity of the data being used has an important effect on the weight given to the information studying it generates. There are many causes that limit data veracity. Data can be biased by the assumptions made by those who collected them. Software bugs can introduce errors and omission in a data set. Abnormalities can reduce data veracity like when two wind speed sensors next to each other report different wind directions. One of the sensors could be failing, but there is no way to determine this from the data itself. Sources can also be of questionable veracity — in a company's social media feed are a series of very

negative reviews. Were they human or bot created? Human error can also be present such as a person signing up to a web service entering their phone number incorrectly. And there are many more ways data veracity can be compromised.

The point of dealing with all this data is to identify useful detail out of all the noise — businesses can find ways to reduce costs, increase speed and efficiency, design new products and brands, and make more intelligent decisions. Governments can find similar benefits in studying the data produced by their citizens and industries.

Here are some examples of current uses of Big Data.

Product Development

Big Data can be used to predict customer demand. Using current and past products and services to classify key attributes, they can then model these attributes' relationships and their success in the market.

Predictive Maintenance

Buried in structured data are indices that can predict the mechanical failure of machine parts and systems. Year of manufacture, make and model, and so on, provide a way to predict future breakdowns. Also, there is a wealth of unstructured data in error messages, service logs, operating temperature, and sensor data. This data, when correctly analyzed, can predict problems before they happen so maintenance can be deployed preemptively, reducing both cost and system downtime.

Customer Experience

Many businesses are nothing without their customers. Yet acquiring and keeping customers in a competitive landscape is difficult and expensive. Anything that can give a business an edge will be eagerly utilized. Using Big Data, businesses can get a much clearer view of the customer experience by examining social media, website visit metrics, call logs, and any other recorded customer interaction to modify and improve the customer experience, all in the interests of maximizing the value delivered in order to acquire and maintain customers. Offers to individual customers can become not only more personalized but more relevant and accurate. By using Big

Data to identify problematic issues, businesses can handle them quickly and effectively, reducing customer churn and negative press.

Fraud & Compliance

While there may be single rogue bad actors out there in the digital universe attempting to crack system security, the real threats are from organized, well-financed teams of experts, sometimes teams supported by foreign governments. At the same time, security practices and standards never stand still but are constantly changing with new technologies and new approaches to hacking existing ones. Big Data helps identify data patterns suggesting fraud or tampering. Aggregation of these large data sets makes regulatory reporting much faster.

Operation Efficiency

Not the sexiest topic, but this is the area in which Big Data is currently providing the most value and return. Data helps companies analyze and assess production systems, examine customer feedback and product returns, and examine a myriad of other business factors to reduce outages and waste, and even anticipate future demand and trends. Big Data is even useful in assessing current decision-making processes and how well they function in meeting demand.

Innovation

Big Data is all about relations between meaningful labels. For a large business, this can mean examining how people, institutions, other entities, and business processes intersect, and use any interdependencies to drive new ways to take advantage of these insights. New trends can be predicted, and existing trends can be better understood. This all leads to understanding what customers actually want and anticipate what they may want in the future. Knowing enough about individual customers may lead to the ability to take advantage of dynamic pricing models. Innovation driven by Big Data is really only limited by the ingenuity and creativity of the people curating it.

Machine Learning is also meant to deal with large amounts of data very quickly. But while Big Data is focused on using existing data to find trends, outliers, and anomalies, Machine Learning uses this same data to “learn” these patterns in order to deal with future data proactively. While Big Data

looks at the past and present data, Machine Learning examines the present data to learn how to deal with the data that will be collected in the future. In Big Data, it is people who define what to look for and how to organize and structure this information. In Machine Learning, the algorithm teaches itself what is important through iteration over test data, and when this process is completed, the algorithm can then go ahead to new data it has never experienced before.

Chapter 11: Machine Learning and Regression

Machine Learning can produce two distinct output types — classification and regression. A classification problem involves an input that requires an output in the form of a label or category. Regression problems, on the other hand, involve an input that requires an output value as a quantity. Let's look at each form in more detail.

Classification Problems

Classification problems are expected to produce an output that is a label or category. That is to say, a function is created from an examination of the input data that produces a discrete output. A familiar example of a classification problem is whether a given email is spam or not spam.

Classification can involve probability, providing a probability estimate along with its classification. 0.7 spam, for example, suggests a 70% chance an existing email is spam. If this percentage meets or exceeds the acceptable level for a spam label, then the email is classified as spam (we have spam folders in our email programs, not 65% spam folders). Otherwise, it is classified as not spam. One common method to determine classification problem's algorithm probability of accuracy is to compare the results of its predictive model against the actual classification of the data set it has examined. On a data set of 5 emails, for example, where the algorithm has successfully classified 4 out of 5 of the emails in the set, the algorithm could be said to have an accuracy of 80%. Of course, in a real-world example, the training data set would be much more massive.

Regression problems

In a regression problem, the expected output is in the form of an unlimited numerical range. The price of a used car is a good example. The input might be the year, color, mileage, condition, etc., and the expected output is a dollar value, such as \$4,500 - \$6,500. The skill (error) of a regression algorithm can be determined using various mathematical techniques. A common skill measure for regression algorithms is to calculate the root mean squared error, RMSE.

Although it is possible to modify each of the above methods to produce each other's result (that is, turn a classification algorithm into a regression algorithm and vice versa), the output requirements of the two define each algorithm quite clearly:

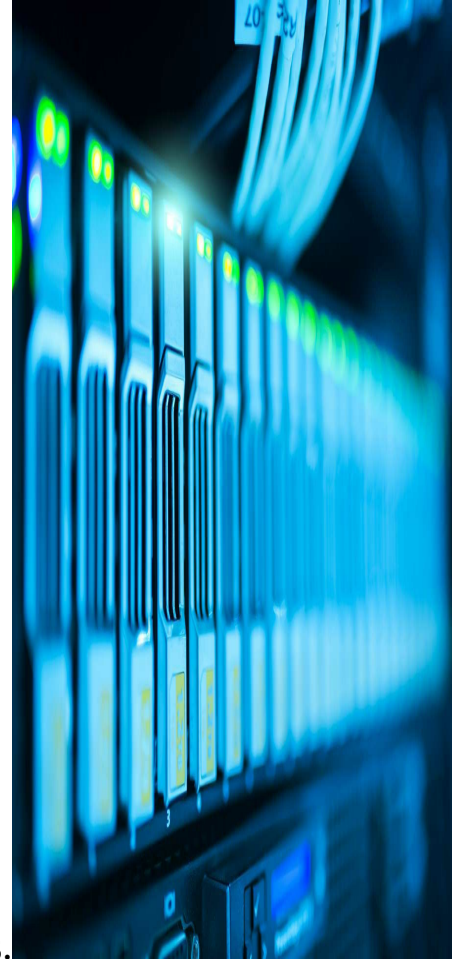
1. Classification algorithms produce a discrete category result which can be evaluated for accuracy, while regression algorithms cannot.
2. Regression algorithms produce a ranging result and can be evaluated using root mean squared error, while classification algorithms cannot.

So, while Machine Learning employs both types of methods for problem-solving (classification and regression), what method is employed for any particular problem depends on the nature of the problem and how the solution needs to be presented.

Chapter 12: Machine Learning and the Cloud

Cloud computing has entered the public consciousness. As a metaphor, it is powerful because it takes a complex activity and turns it into a simple idea. But cloud computing is a complex and expensive undertaking. It involves the integration of hundreds, thousands, or even tens of thousands of computer servers in a massive data center (a single Amazon data center can host as many as 80,000 servers). This “raw iron” serves as the backbone of the cloud service. The servers operate virtual machine managers called hypervisors, which can be either software or hardware that uses the raw processing power of the existing data center servers to simulate other software, be that individual programs or entire operating systems.

The advantage of cloud computing is the fact the cost associated with all that hardware in the data center is absorbed by the company providing the cloud services. This includes the building and infrastructure to operate the hardware, the electricity to power it, and the staff to maintain, service,



update, and repair hardware and software systems.

These companies monetize their hardware and virtualized software by selling it on demand to whoever wants to use it, be that other companies, governments, universities, or individuals. And even though the cloud service is more expensive than purchasing and using their own servers, the advantage is in the on-demand nature of the service. A company may have a spike in its retail website use around Christmas, for example, where demand surges and would overwhelm the servers that are able to operate without issue in the rest of the year. Instead of having to purchase more hardware to manage this spike in use, hardware that would remain essentially dormant the rest of the year (though still consuming electricity, requiring maintenance, and updates by trained staff, and so on), this company could instead acquire a cloud services account and “virtualize” part of its website business using load balancing to make sure its servers are not overwhelmed. If demand is even more than expected, in a matter of moments, more servers can be spun up to

carry the demand. As traffic decreases late at night, or for whatever reason, these extra virtual servers can be shut down, no longer costing the company money to operate.

Previously, we've discussed the forces that allowed Machine Learning to come into its own, the historical development of the mathematical algorithms being used, the rise of very large, numerous sets of data, and the increasing power of computer processors. But even as these factors merged to make Machine Learning viable, it is extremely expensive. There are enormous resources required, from qualified software developers who know the math behind Machine Learning algorithms, to gaining access to large data sets either through business operations or purchase, to being able to afford the massive processing power to crunch all that data and allow an algorithm to learn. Access to Machine Learning has for the most part been only for governments, research organizations like universities, and large businesses.

The Cloud promises to change this. As Machine Learning systems are developed and refined by researchers and business ahead of the curve in the industry, they can often end up being available to consumers in cloud-based services provided by companies like Amazon AWS, Google Cloud, and Microsoft Azure. Because these services can provide these algorithms on demand, the cost per use for them can be quite low. At the same time, cloud-based storage has become very reasonably priced, meaning those massive data sets needed for training and testing Machine Learning algorithms can be stored in the cloud as well, reducing cost for storage, the infrastructure required to carry the bandwidth needed to move such massive amounts of data, and the time required to make such transfers. The software and data can reside in the same cloud account, providing instant access, without any bandwidth costs.

Companies using cloud computing to implement Machine Learning do not need an IT department capable of creating and managing the AI infrastructure, as this is taken care of by the companies offering the cloud services.

Just as cloud-based services offer SaaS (software as a service) solutions, Machine Learning cloud services offer SDKs (software developer kits) and APIs (application programming interfaces) to embed Machine Learning

functions into applications. These connections offer support for most programming languages, meaning developers working for the companies using the cloud-based solutions do not need to learn any new languages to harness the Machine Learning infrastructure. Developers can harness the power of Machine Learning processes directly in their applications, which is where the most value lies because most real-world use of Machine Learning systems today are transaction and operations-focused such as real-time loan applications, fraudulent transactions, mapping, and route planning, facial recognition, voice authorization, and so on.

Benefits of Cloud-Based Machine Learning

It is possible to use one of the free open source Machine Learning frameworks such as CNTK, TensorFlow, and MXNet to create your own AI solutions, but even though the frameworks are free, the barriers to do it yourself approach can be prohibitive. We've touched on them above — overall cost for hardware and hardware maintenance, the cost for software maintenance, and the cost of acquiring and maintaining a team of AI specialists capable of designing and maintaining Machine Learning systems.

Cloud-based Machine Learning projects make it easy for a company or organization to use limited levels of resources during experimentation and training while providing a seamless ability to scale up for production and, in the future, as demand increases. Whereas with an in-house Machine Learning option, after training has taken place and the algorithm is ready for production deployment, an entire infrastructure must be created around the software for outside systems to use it. With a cloud solution, production is often as simple as a few mouse clicks away.

Amazon AWS, Google Cloud Platform, and Microsoft Azure have many Machine Learning services that do not require knowledge of Artificial Intelligence, theories of Machine Learning, or even a data science team.

At this point, the big cloud service providers provide two basic kinds of Machine Learning offerings — general and specific. A specific offering, for example, is Amazon's Rekognition, an image-recognition Machine Learning implementation that can be run with a single command. On the other hand, if what you require is not available from one of these specific user options, all three services offer more generalized Machine Intelligence solutions requiring the user to create custom code and to be run on general-purpose services.

There has been an attempt to make services that offer more general-purpose Machine Learning services that are simpler to use, but as with many attempts at generalized software frameworks, the market hasn't embraced the concept because while being easier to use, the simplicity of the interface means it is

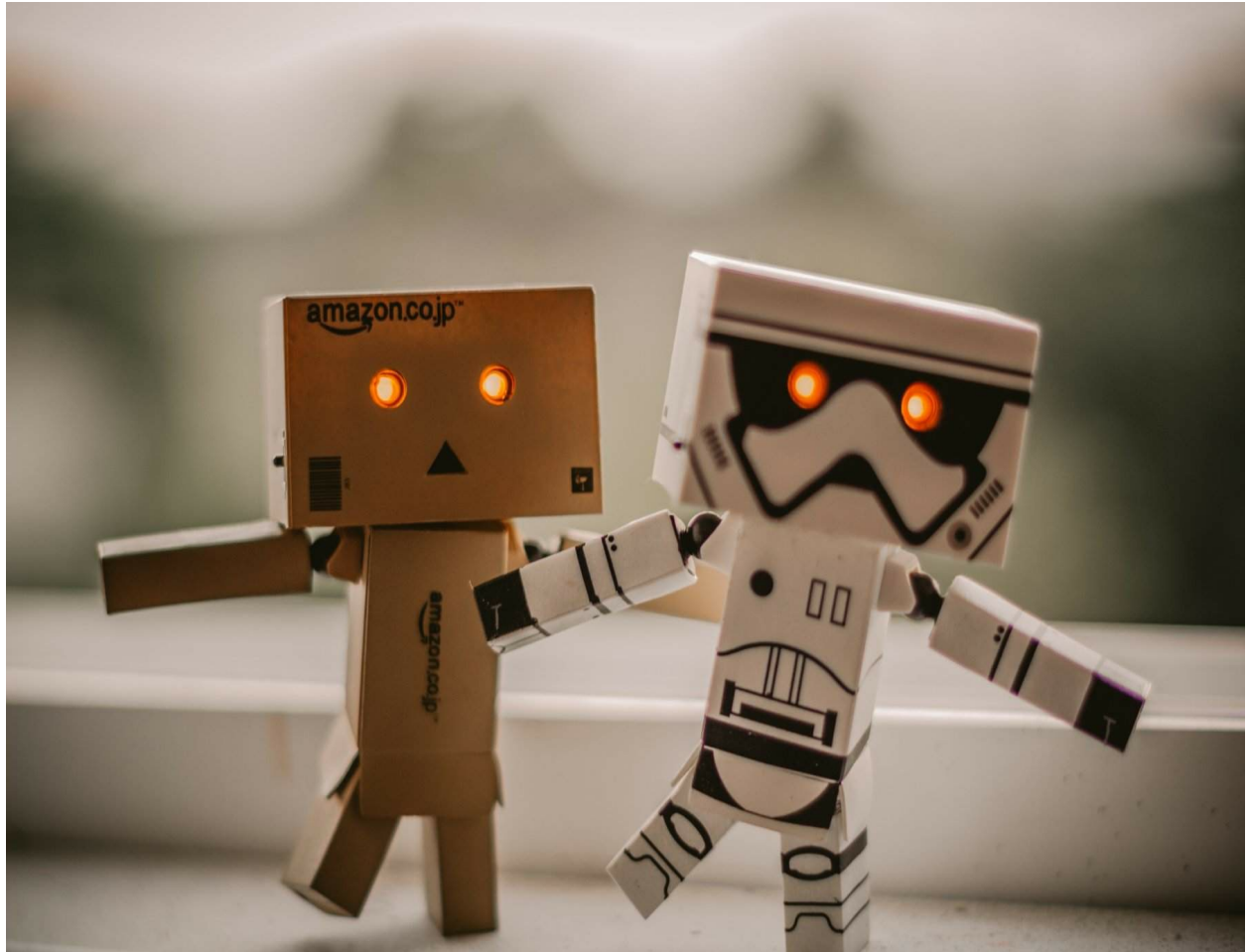
not possible to get the custom requirements customers are looking for.

Chapter 13: Machine Learning and the Internet of Things (IoT)

Perhaps the first Internet of Things was at Carnegie Mellon University in 1982 when a Coke machine was connected to the internet so it could report on its current stock levels, as well as the temperature of newly-loaded drinks.

But at that time, computer miniaturization had not progressed to the point it has today, so it was difficult for people at that time to conceive of very small devices connecting to the internet. Also, consumer wireless internet would not be available for another 15 years, meaning any internet connected device had to be fairly large and wired to its connection.

Today, the Internet of Things (IoT) refers to all the Internet-connected devices in the world. Some are obvious like your smartphone, and some are not so obvious like a smart thermostat in your house. The term also applies to internet-connected devices that are part of a device like a sensor on an industrial robot's arm or the jet engine from a commercial jetliner.



A relatively simple explanation, but it is hard to conceive just how many IoT devices are around us, never mind how many will be in the near future. The following is a breakdown of some of the Internet of Things devices currently in use, which is often broken down into the following categories:

Consumer Applications

The number of Internet of Things devices created for consumer use is expanding rapidly. From smartphones to connected vehicles to wearable health technology and health monitoring devices, the Internet of Things is entering our lives at a rapid pace. Many of these applications are the result of simple programming methods, like those devised in Python.

Smart Home

Home automation is the concept of having your house manage its resources for you. These resources include lighting, fridge, air conditioning, security, media, and so on. The Internet of Things is critical to this philosophy because the means to manage and automate home resources come through the ability of the, until now, separate devices being able to connect to the internet and possibly to each other. These connections allow the owner to adjust the lighting or temperature or view areas of the home when outside the home. At the same time, these devices are able to alert the owner, sending a warning that an external perimeter has been breached and offering a real-time video display of the culprit – more likely a raccoon than a burglar, but in either case, the devices can provide a feeling of control. Another example might be when your smart fridge notices your milk is 90% empty and automatically adds it to your electronic shopping list.

Clearly, we could come up with the examples of many more devices currently being used, as well as all those not yet created that will be coming in the future, that employ various kinds of Machine Learning. The security system above needs to determine something is moving in the backyard. Perhaps more advanced Machine Learning would even be able to distinguish a person from a small animal and not bother you at all. At the same time, it could possibly use facial recognition to determine it is only your neighbor who came around to fetch a Frisbee accidentally thrown over the adjoining fence.

The smart fridge not only needs to be able to classify the items it is holding, but to judge when enough of an item has been consumed to justify ordering

more – clearly a Machine Learning problem since 70% consumed might mean less than a day left for a family of three, but 2 days’ supply for a single person. These details will be learned, as a Machine Learning algorithm in your fridge first observes the occupants using the fridge and determines the correct quantities for reorder. When another person moves back in for the summer after college, a Machine Learning algorithm can observe and adjust its estimates accordingly. Over time, these predictive models can aggregate their data, allowing new instances of the algorithm to auto-correct for added or removed residents, based on the data sets of previous move ins and move outs.

Elder/Disabled Care

One of the Smart Home’s strongest selling features is its ability to help care for people with disabilities. Among assistive technologies available is voice control to activate/deactivate or set features around the house — lighting, temperature, etc., and the ability to identify medical emergencies such as seizures or falls. No doubt, these features will expand, including an option for diagnosis and treatment suggestions, as well as the ability to interface with emergency services and local hospitals to notify and assist in dealing with medical emergencies.

Machine Learning in the home is already available in terms of voice recognition and fall monitoring. It will be instrumental in more extensive medical monitoring, including the ability to “read” emotions and monitor our health via external and even internal sensors.

Here again, we see how the integration of the IoT and the health care system can provide a benefit, but behind many of these qualities of life-improving devices is the predictive and anomaly detection capacity of Machine Learning algorithms.

Commercial Applications

Healthcare

We are entering the world of “Smart Healthcare,” where computers, the internet, and artificial intelligence are merging to improve our quality of life. The Internet of Medical (or Health) Things (IoHT) is a specific application of the Internet of Things designed for health and medically related purposes. It is leading to the digitization of the healthcare system, providing connectivity between properly equipped healthcare services and medical resources.

Some Internet of (Heath) Things applications enable remote health monitoring and operate emergency notification systems. This can range from blood pressure monitoring and glucose levels to the monitoring of medical implants in the body. In some hospitals, you will find “smart beds” that can detect patients trying to get up and adjust themselves to maintain healthy and appropriate pressure and support for the patient, without requiring the intervention of a nurse or other health professional. One report estimated these sorts of devices and technology could save the US health care system \$300 billion in a year in increased revenue and decreased costs. The interactivity of medical systems has also led to the creation of “m-health,” which is used to collect and analyze information provided by different connected resources like sensors and biomedical information collection systems.

Rooms can be equipped with sensors and other devices to monitor the health of senior citizens or the critically ill, ensuring their well-being, as well as monitor that treatments and therapies are carried out to provide comfort, regain lost mobility, and so on. These sensing devices are interconnected and can collect, analyze, and transfer important and sensitive information. In-home monitoring systems can interface with a hospital or other health-care monitoring stations.

There have been many advances in plastics and fabrics allowing for the creation of very low-cost, throw away “wearable” electronic IoT sensors. These sensors, combined with RFID technology, are fabricated into paper or

e-textiles, providing wirelessly-powered, disposable sensors.

Combined with Machine Learning, the health care IoHT ecosphere around each one of us can improve quality of life, guard against drug errors, encourage health and wellness, and respond to and even predict responses by emergency personnel. As a collection of technology and software, this future “smart” health care will cause a profound shift in medical care, where we no longer wait for obvious signs of illness to diagnose, but instead use the predictive power of Machine Learning to detect anomalies and predict future health issues long before human beings even know something might be wrong.

Transportation

The IoT assists various transportation systems in the integration of control, communications, and information processing. It can be applied throughout the entire transportation system — drivers, users, vehicles, and infrastructure. This integration allows for inter and even intra-vehicle communication, logistics and fleet management vehicle control, smart traffic control, electronic toll collection systems, smart parking, and even safety and road assistance.

In case of logistics and fleet management, an IoT platform provides continuous monitoring of cargo and asset locations and conditions using wireless sensors and send alerts when anomalies occur (damage, theft, delays, and so on). GPS, temperature, and humidity sensors can return data to the IoT platform where it can be analyzed and set on to the appropriate users. Users are then able to track in real-time the location and condition of vehicles and cargo and are then able to make the appropriate decisions based on accurate information. IoT can even reduce traffic accidents by providing drowsiness alerts and health monitoring for drivers to ensure they do not drive when they need rest.

As the IoT is integrated more and more with vehicles and the infrastructure required to move these vehicles around, the ability for cities to control and alleviate traffic congestion, for businesses to control and respond to issues with transportation of their goods, and for both of these groups to work together, increases dramatically. In unforeseen traffic congestion due to an

accident, for example, sensitive products (a patient in an ambulance on the way to Emergency or produce or other time-sensitive items) could be put in the front of the queue to ensure they are delayed as little as possible, all without the need for traffic direction by human beings.

The potential rewards of such an integrated system are many, perhaps only limited by our imagination. Using IoT enabled statistics from traffic will allow for optimum traffic routing, which in turn will reduce travel time and CO2 emissions. Smart stop lights and road signs with variable-speed and information displays will communicate more and more with the onboard systems of vehicles, providing routing information to reduce travel time. And all of this technology is made possible by dedicated Machine Learning algorithms with access to the near endless flow of data from thousands and thousands of individual IoT sensors.

Building/Home Automation

As discussed above, IoT devices can be used in any kind of building, where they can monitor and control the electrical, mechanical, and electronic systems of these buildings. The advantages identified are:

- By integrating the internet with a building's energy management systems, it is possible to create "smart buildings" where energy efficiency is driven by IoT.
- Real-time monitoring of buildings provides a means for reducing energy consumption and the monitoring of occupant behavior.
- IoT devices integrated into buildings provide information on how smart devices can help us understand how to use this connectivity in future applications or building designs.

When you read "smart" or "real-time" above, you should be thinking Machine Learning because it is these algorithms that can tie all of the sensor input together into something predictive and intelligent.

Industrial Applications

As we've mentioned above, programming and machines that are commonly used by society are on an upswing. But what about industrial applications? You've probably had a friend or family member who suffered from a disease or condition treatable only in the hospital. These types of diseases are best understood through programmed technology.

Other media, such as university programming, is essential to the research of various scientists. Data acquisition is far from a thing of the past, but the method by which we interpret and record information has improved exponentially. These are just a few means by which we program equipment and provide data for people all over the world.

Manufacturing

Manufacturing equipment can be fitted with IoT devices for sensing, identification, communication, actuation monitoring, processing, and networking, providing seamless integration between the manufacturing and control systems. This integration has permitted the creation of brand-new business and marketing options in manufacturing. This network control and management of manufacturing process control, asset and situation management, and manufacturing equipment set the IoT in the industrial application and smart manufacturing space, allowing features such as^[7]:

- Rapid manufacturing of new products.
- Real-time product and supply chain optimization by networking machine sensors and their control systems together.
- Allowing dynamic responses to product or market demand

The above is achieved by networking manufacturing machinery, control systems, and sensors together.

The IoT can be applied to employ digital control systems for automating process controls, manage service information systems in the optimization of plant safety and security, and maintain and control operator tools.

Machine Learning, when integrated into these networked systems, allows asset management and surveillance to maximize reliability by employing predictive maintenance and statistical evaluation. Real-time energy optimization is possible through the integration of the various sensor systems.

Machine Learning allows IoT to maximize reliability through asset management using predictive maintenance, statistical evaluation, and measurements.

The manufacturing industry has its own version of the Internet of Things, the industrial Internet of Things (IIoT). The estimated effect IIoT could have on the manufacturing sector is truly astounding. Growth due to the integration of the Internet of Things with manufacturing is estimated at a \$12 trillion dollar increase in global GDP by 2030.

Data acquisition and device connectivity are imperative for IIoT. This is not the goal, but rather a necessary condition for developing something much greater. At the core of this will be interfacing Machine Learning algorithms with local IIoT devices. This has already been done in some instances with predictive maintenance, and this success will hopefully pave the way for new and much more powerful ways to allow these intelligent maintenance systems to reduce unforeseen downtime and provide increased productivity. This term refers to the interface between human beings and the cyber world, and this system will eventually allow data collected from it into actionable information, decreasing costs, increasing safety, and boosting efficiency. As with so many implementations of Machine Learning in our world, it would seem the only limit to the advantages it can provide in the manufacturing sector relies on the limits of our imagination to take advantage of it.

A real-world example of this integrated approach was conducted by the University of Cincinnati in 2014. Industrial band-saws have belts that degrade and eventually, if not caught by routine inspection, break, presenting a hazard to the operators. In this experiment, a feedback system was created in which Industrial Internet of Things sensors monitored the status of the belt and returned this data to a predictive system, which could alert users when the band was likely to fail and when the best time to replace it would be. This system showed it was possible to save on costs, increase operator safety, and improve the user experience by integrating IIoT devices with Machine

Learning algorithms.

Agriculture

The IoT allows farmers to collect data on all sorts of parameters, things like temperature, humidity, wind speed/direction, temperature, soil composition, and even pest infestations.

The collection of this data when combined with Machine Learning algorithms, allows farmers to automate some farm techniques such as making informed decisions to improve crop quality and quantity, minimize risk, reduce required crop management efforts, and minimize waste. Farmers can monitor soil temperature and moisture and use this data to decide on an optimum time to water or fertilize fields. In the future, Machine Learning can use historical and current weather data in combination with data returned from the Internet of Things devices embedded in the farm to further maximize these decision-making processes.

An example of a real-world combination of Machine Learning and IoT in agriculture took place in 2018, when Toyota Tsusho (member of the Toyota group which is parent to, among many other things, Toyota Motors) partnered with Microsoft and created fish farming tools employing the Microsoft Azure Internet of Things application suite for water management. The water pump in this facility uses Machine Learning systems provided by Azure to count fish on a conveyor belt and use this result to determine how effective the water flow was in the system. This one system employed both the Microsoft Azure IoT Hub and Microsoft Azure Machine Learning platforms.

Infrastructure Applications

The Internet of Things can be used for control and monitoring of both urban and rural infrastructure, things like bridges, railways, and off and on-shore wind farms. The IoT infrastructure can be employed to monitor changes and events in structural conditions that might threaten safety or increase risks to users.

The construction industry can employ the Internet of Things and receive benefits like an increase in productivity, cost savings, paperless workflows, time reduction, and better-quality work days for employees, while real-time monitoring and data analytics can save money and time by allowing faster decision-making processes. Coupling the Internet of Things with Machine Learning predictive solutions allows for more efficient scheduling for maintenance and repair, as well as allowing for better coordination of tasks between users and service providers. Internet of Things deployments can even control critical infrastructure by allowing bridge access control to approaching ships, saving time and money. Large-scale deployment of IoT devices for the monitoring and operation of infrastructure will probably improve the quality of service, increase uptime, reduce the costs of operations, and increase incident management and emergency response coordination. Even waste-management could benefit from IoT deployments, allowing for the benefits of optimization and automation.

City Scale Uses

The Internet of Things has no upper limit to what it can encompass. Today, buildings and vehicles are routine targets for the Internet of Things integration, but tomorrow, it will be cities that are the wide-scale target of this digital revolution. In many cities around the world, this integration is beginning with important effects on utilities, transportation, and even law enforcement. In South Korea, for example, an entire city is being constructed as the first of its kind, a fully equipped and wired city. Much of the city will be automated, requiring very little or even no human intervention. Songdo is a nearly finished construction.

The city of Santander in Spain is taking a different approach. It does not have the benefit of being built from scratch. Instead, it has produced an app that is connected with over 10,000 sensors around the city. Things like parking search, a digital city agenda, environmental monitoring, and more have been integrated by the Internet of Things. Residents of the city download an app to their smartphones in order to access this network.

In San Jose California, an Internet of Things deployment has been created with several purposes — reducing noise pollution, improving air and water quality, and increasing transportation efficiency. In 2014, the San Francisco Bay Area partnered with Sigfox, a French company to deploy an ultra-narrow band wireless data network, the first such business-backed installation in the United States. They planned to roll out over 4000 more base stations to provide coverage to more than 30 U.S. cities.

In New York, all the vessels belonging to the city were connected, allowing 24/7 monitoring. With this wireless network, NYWW can control its fleet and passengers in a way that was simply not possible to imagine even a decade ago. Possible new applications for the network might include energy and fleet management, public Wi-Fi, security, paperless tickets, and digital signage among others.

Most of the developments listed above will rely on Machine Learning as the intelligent back-end to the forward-facing Internet of Things used to monitor and update all the components of any smart city. It would seem the next step, after the creation of smart cities through the integration of Internet of Things sensors and monitors with Machine Learning algorithms, would be larger, regional version.

Energy Management

So many electrical devices in our environment already boast an internet connection. In the future, these connections will allow communication with the utility from which they draw their power. The utility, in turn, will be able to use this information to balance energy generation with load usage and to optimize energy consumption over the grid. The Internet of Things will allow users to remotely control (and schedule) lighting, HVAC, ovens, and so on. On the utility's side, aspects of the electrical grid, transformers, for example,

will be equipped with sensors and communications tools, which the utility will use to manage the distribution of energy.

Once again, the power of Machine Learning will provide the predictive models necessary to anticipate and balance loads, aided by the constant flow of information from the Internet of Things devices located throughout the power grid. At the same time, these IoT monitors will also provide service levels and other information, so artificial intelligence systems can track and identify when components are reaching the end of life and need repair or replacement. No more surprise power outages, as components getting close to failure will be identified and repaired before they can threaten the power grid.

Environmental Monitoring

IoT can be used for environmental monitoring and usually, that application means environmental protection. Sensors can monitor air and water quality, soil and atmospheric conditions, and can even be deployed to monitor the movement of wildlife through their habitats. This use of the Internet of Things over large geographic areas means it can be used by tsunami or earthquake early-warning systems, which can aid emergency response services in providing more localized and effective aid. These IoT devices can include a mobile component. A standardized environmental Internet of Things will likely revolutionize environmental monitoring and protection.

Here, we can begin to see the geographic scalability of Machine Learning as it interfaces with the Internet of Things. While the use in cities can be impressive due to the number of sensors and devices deployed throughout them, in rural or natural environmental settings, the density of IoT devices drops sharply, yet these huge geographical areas can come under the scrutiny of Machine Learning algorithms in ways yet to be conceived.

Trends in IoT

The trend for the Internet of Things is clear — explosive growth. There were an estimated 8.4 billion IoT devices in 2017, a 31% increase from the year before. By 2020, worldwide estimates are that there will be 30 billion devices. At this point, the market value of these devices will be over \$7 trillion. The amount of data these devices will produce is staggering. And when all this data is collected and ingested by Machine Learning algorithms, control, and understanding of so many aspects of our lives, our cities, and even our wildlife will increase dramatically.

Intelligence

Ambient Intelligence is an emerging discipline in which our environment becomes sensitive to us. It was not meant to be part of the Internet of Things, but the trend seems to be that Ambient Intelligence and autonomous control are two disciplines into which the Internet of Things will make great inroads. Already research in these disciplines is shifting to incorporate the power and ubiquity of the IoT.

Unlike today, the future Internet of Things might be comprised of a good number of non-deterministic devices, accessing an open network where they can auto-organize. That is to say, a particular IoT device may have a purpose, but when required by other Internet of Things devices, it can become part of a group or swarm to accomplish a collective task that overrides its individual task in priority. Because these devices will be more generalized with a suite of sensors and abilities, these collectives will be able to organize themselves using the particular skills of each individual to accomplish a task, before releasing its members when the task is complete so they can continue on with their original purpose. These autonomous groups can form and disperse depending on the priority given to the task, and considering the circumstances, context, and the environment in which the situation is taking place.

Such collective action will require coordination and intelligence, which of course will need to be provided, at least in part, by powerful Machine

Learning algorithms tasked with achieving these large-scale objectives. Even if the Internet of Things devices that are conscripted to achieve a larger goal have their own levels of intelligence, it will be assisted by artificial intelligence systems able to collate all the incoming data from the IoT devices responding to the goal and finding the best approach to solve any problems. Although swarm intelligence technology will also likely play a roll (see later in the book in the chapter on The Swarm), swarms cannot provide the predictive ability that crunching massive data sets by Machine Learning backed by powerful servers have.

Architecture

The system architecture of IoT devices is typically broken down into three main tiers. Tier one includes devices, Tier two is the Edge Gateway, and Tier three is the Cloud.

Devices are simply network-connected things with sensors and transmitters using a communication protocol (open or proprietary) to connect with an Edge Gateway.

The Edge Gateway is the point where incoming sensor data is aggregated and processed through systems to provide functionality including data pre-processing, securing cloud connections, and running WebSockets or even edge analytics.

Finally, Tier three includes the applications based in the cloud built for the Internet of Things devices. This tier includes database storage systems to archived sensor data. The cloud system handles communication taking place in all tiers and provides features like event queuing.

As data is passed from the Internet of Things, a new architecture emerges, the web of things, the application layer which processes the disparate sensor data from the Internet of Things devices into web applications, driving innovation in use-cases.

It is no great leap to assume this architecture does, and will even more in the future, require massive scalability in the network. One solution being deployed is fog (or edge) computing, where the edge network, tier two above, takes over many of the analytical and computational tasks, sorting the incoming data and only providing vetted content to the tier three cloud layer.

This reduces both latency and bandwidth requirements in the network. At the same time, if communications are temporarily broken between the cloud and edge layers, the edge layer's fog computing capabilities can take over to run the tier one devices until communications can be re-established.

Complexity and Size

Today's Internet of Things is more a collection of proprietary devices, protocols, languages, and interfaces. In the future, as there is more standardization deployed, this collection will be studied as a complex system, given its fast numbers of communication links, autonomous actors, and its capacity to constantly integrate new actors. Today, however, most elements of the Internet of Things do not belong to any massive group of devices. Devices in a smart home, for example, do not communicate with anything other than the central hub of the home. These subsystems are there for reasons of privacy, reliability, and control. Only the central hub can be accessed or controlled from outside the home, meaning all sensor data, positions, and the status of the devices inside are not shared. Until such time as there is an assurance of security from more global networks about access and control of these private networks of IoT devices, and at the same time, until non-standard architectures are developed to allow the command and control of such vast numbers of internet connected devices, the IoT will likely continue as a collection of non-communicating mini-networks. SND, Software Defined Networking, appears promising in this area as a solution that can deal with the diversity and unique requirements of so many IoT applications.

For Machine Learning to take advantage of all this sensor data, ways will need to be devised to collect it into training and data sets that can be iterated over, allowing the software to learn about and be able to predict important future states of the IoT network. For the time being, unfortunately, most IoT sensor data produced in these private networks is simply being lost.

Space

Today, the question of an Internet of Thing's location in space and time has not been given priority. Smartphones, for example, allow the user to opt out of allowing an application gaining access to the phone's capacity for

geolocation. In the future, the precise location, position, size, speed, and direction of every IoT device on the network will be critical. In fact, for IoT devices to be interconnected and provide coordinated activity, the current issues of variable spatial scales, indexing for fast search and close neighbor operations, and indeed just the massive amount of data these devices will provide all have to be overcome. If the Internet of Things is to become autonomous from human-centric decision-making, then in addition to Machine Learning algorithms being used to command and coordinate them, space and time dimensions of IoT devices will need to be addressed and standardized in a way similar to how the internet and Web have been standardized.

Jean-Louis Gassée and a “basket of remotes”

Gassée worked at Apple from 1981 to 1990 and after that was one of the founders of Be Incorporated, creator of the BeOS operating system. He describes the issue of various protocols among Internet of Things providers as the “basket of remotes” problem. As wireless remote controls gained popularity, consumers would find they ended up with a basket full of remotes — one for the TV, one for the VCR, one for the DVD player, one for the stereo, and possibly even a universal remote that was supposed to replace all the others but often failed to match at least one device. All these devices used the same technology, but because each manufacturer used a proprietary language and/or frequency to communicate with their appliance, there was no way to easily get one category of device to speak the same language as the others. In the same way, what Gassée sees happening is, as each new deployment of some Internet of Things tech reaches the consumer market, it becomes its own remote and base station, unable to communicate with other remotes or with the other base station to which other groups of IoT tech can communicate. If this state of proprietary bubbles of the Internet of Things collections is not overcome, many of the perceived benefits of a vast IoT network will never be realized.

Security

Just as security is one of the main concerns for the conventional internet, security of the Internet of Things is a much-discussed topic. Many people are concerned that the industry is developing too rapidly and without an

appropriate discussion about the security issues involved in these devices and their networks. The Internet of Things, in addition to standard security concerns found on the internet, has unique challenges — security controls in industry, Internet of Things businesses processes, hybrid systems, and end nodes.

Security is likely the main concern over adopting Internet of Things tech. Cyber-attacks on components of this industry are likely to increase as the scale of IoT adoption increases. And these threats are likely to become physical, as opposed to merely a virtual threat. Current Internet of Things systems have many security vulnerabilities including a lack of encrypted communication between devices, weak authentication (many devices are allowed to run in production environments with default credentials), lack of verification or encryption of software updates, and even SQL injection. These threats provide bad actors the ability to easily steal user credentials, intercept data, and collect Personally Identifiable Information, or even inject malware into updated firmware.

Many internet-connected devices are already spying on people in their own homes, including kitchen appliances, thermostats, cameras, and televisions. Many components of modern cars are susceptible to manipulation should a bad actor gain access to the vehicle's onboard systems including dashboard displays, the horn, heating/cooling, hood and trunk releases, the engine, door locks, and even braking. Those vehicles with wireless connectivity are vulnerable to wireless remote attacks. Demonstration attacks on other internet-connected devices have been made as well, including cracking insulin pumps, implantable cardioverter defibrillators, and pacemakers. Because some of these devices have severe limitations on their size and processing power, they may be unable to use standard security measures like strong encryption for communication or even employing firewalls.

Privacy concerns over the Internet of Things have two main thrusts — legitimate and illegitimate uses. In legitimate uses, governments and large corporations may set up massive IoT services which by their nature collect enormous amounts of data. To a private entity, this data can be monetized in many different ways, with little or no recourse for the people whose lives and activities are swept up in the data collection. For governments, massive data

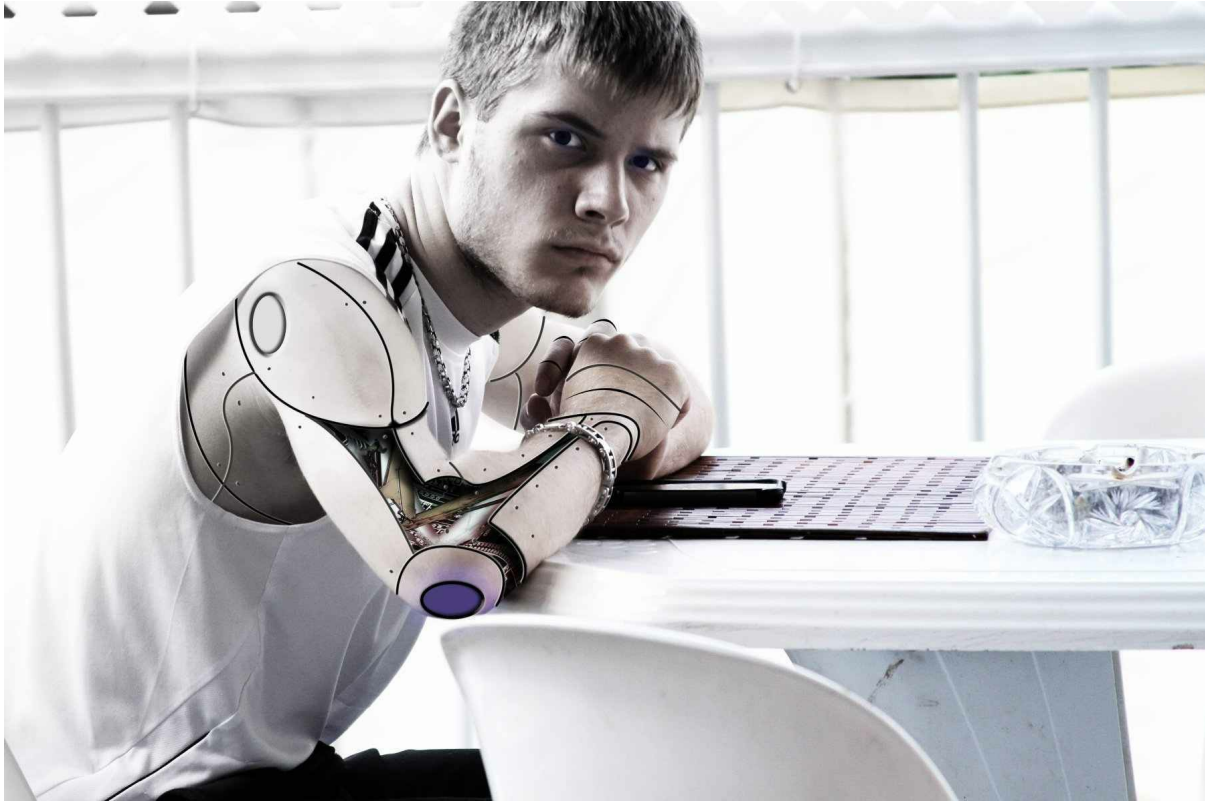
collection from the Internet of Things networks provide the data necessary to provide services and infrastructure, to save resources and reduce emission, and so on. At the same time, these systems will collect enormous amounts of data on citizens, including their locations, activities, shopping habits, travel, and so on. To some, this is the realization of a true surveillance state. Without a legal framework in place to prevent governments from simply scooping up endless amounts of data to do with as they wish, it is difficult to refute this argument.

Illegitimate uses of the massive Internet of Things networks include everything from DDOS (distributed denial of service) attacks through malware attacks on one or more of the IoT devices on the network. Even more worrying, security vulnerabilities in even one device on an Internet of Things network can, by nature of the fact it is capable of full communication with all devices nearby because it has access to the encryption requirements to present itself as a legitimate device on the network, means an infected device may not only provide its illegitimate host with the data it provides, but metadata of other devices in the network, and possibly even access to the edge systems themselves.

In 2016, a DDOS attack powered by an Internet of Things device compromised by a malware infection led to over 300,000 device infections and brought down both a DNS provider and several major websites. This Mirai Botnet was able to single out for attack devices that consisted mostly of IP cameras, DVRs, printers, and routers.

While there are several initiatives being made to increase security in the Internet of Things marketplace, many argue that government regulation and inter-governmental cooperation around the world must be in place to ensure public safety.

Chapter 14: Machine Learning and Robotics



First, we need to define a robot. They are machines, often programmable by a computer, that are able to carry out a series of complex actions without intervention. A robot can have its controls systems embedded or be controlled by an external device. In popular literature and film, many robots are designed to look like people, but in fact, they are usually designed to perform a task, and that requirement determines how they appear.

Robots have been with us for almost as long as computers. George Devol invented the first digitally operated and programmable one in 1954, called Unimate. In 1961, General Motors bought it to use for lifting hot metal die castings. And like computers, robots have changed our society. Their strength, agility, and ability to continue to execute the same repetitive tasks perfectly have proved an enormous benefit. And while they did cause some serious disruption to the manufacturing industries, putting many people out of work, their ascension in our societies has provided far more employment

opportunities than they have taken.

Robots in current use can be broken down into several categories:

Industrial Robots/Service Robots

These robots are probably familiar. You have likely seen them on the television or streaming video of automated factories. They usually consist of an “arm” with one or more joints, which ends with a gripper or manipulating device. They first took hold in automobile factories. They are fixed in one location and are unable to move about. Industrial robots will be found in manufacturing and industrial locations. Service robots are basically the same in design as industrial robots but are found outside of manufacturing concerns.

Educational Robots

These are robots used as teacher aids or for educational purposes on their own. As early as the 1980s, robots were introduced in classrooms with the turtles, which were used in classrooms where students could train them using the Lego programming language. There are also robot kits available for purchase like the Lego Mindstorm.

Modular Robots

Modular robots consist of several independent units that work together. They can be identical or have one or more variations in design. Modular robots are able to attach together to form shapes that allow them to perform tasks. The programming of modular robotic systems is of course, more complex than a single robot, but ongoing research in many universities and corporate settings is proving that this design approach is superior to single large robots for many types of applications. When combined with Swarm Intelligence (see below), modular robots are proving quite adept at creative problem-solving.

Collaborative Robots

Collaborative robots are designed to work with human beings. They are mostly industrial robots that include safety features to ensure they do not harm anyone as they go about their assigned tasks. An excellent example of this kind of collaborative robot is Baxter. Introduced in 2012, Baxter is an industrial robot designed to be programmed to accomplish simple tasks but is

able to sense when it comes into contact with a human being and stops moving.

Of course, all the examples above do not require any artificial intelligence. When robots are coupled with machine learning, researchers use the term “Robotic Learning.” This field has a contemporary impact in at least four important areas:

Vision

Machine Learning has allowed robots to sense their environment visually and to make sense of what they are seeing. New items can be understood and classified without the need to program into the robot ahead of time what it is looking at.

Grasping

Coupled with vision, Machine Learning allows robots to manipulate items in their environment that they have never seen before. In the past, in an industrial factory, each time a robot was expected to interact with a different-shaped object, it would have to be programmed to know how to manipulate this new object before it could be put to work. With Machine Learning, the robot comes equipped with the ability to navigate new item shapes and sizes automatically.

Motion Control

With the aid of Machine Learning, robots are able to move about their environments and avoid obstacles in order to continue their assigned tasks.

Data

Robots are now able to understand patterns in data, both physical and logistical, and act accordingly on those patterns.

Examples of Industrial Robots and Machine Learning

One example of the benefit of applying Machine Learning to robots is of an industrial robot which receives boxes of frozen food along a conveyor. Because it is frozen, these boxes often have frost, sometimes quite a lot of frost. This actually changes the shape of the box randomly. Thus, a traditionally-trained robot with very little tolerance for these shape changes would fail to grasp the boxes correctly. With Machine Learning algorithms, the robot is now able to adapt to different shapes, random as they are and in real time, and successfully grasp the boxes.

Another industrial example includes a factory with over 90,000 different parts. It would not be possible to teach a robot how to manipulate these many items. With Machine Learning, the robot is able to be fed images of new parts it will be dealing with, and it can determine its own method to manipulate them.

In 2019, there will be an estimated 2.6 million robots in service on the planet. That's up a million from 2015. As more and more robots are combined with Machine Learning algorithms, this number is sure to explode.

Neural Networks with Scikit-learn

Neural networks are a machine learning framework that tries to mimic the way the natural biological neural networks operate. Humans have the capacity of identifying patterns with a very high degree of accuracy. Anytime you see a cow, you can immediately recognize that it is a cow. This also applies to when you see a goat. The reason is that you have learned over a period of time how a cow or a goat looks like and what differentiates between the two.

Artificial neural networks refer to computation systems that try to imitate the capabilities of human learning via a complex architecture that resembles the nervous system of a human being.

Chapter 15: Machine Learning and Swarm Intelligence

Swarm Intelligence (SI) is defined as collaborative behavior, natural or artificial, of decentralized, self-organized systems. That is, Swarm Intelligence can refer to an ant colony or a “swarm” of autonomous mini-drones in a lab.

In artificial intelligence, a swarm is typically a collection of agents that interact with each other and their environment. The inspiration for Swarm Intelligence comes from nature, from the collaboration of bees to the flocking of birds to the motions of herd animals, groups of animals that appear to act intelligently even when no single individual has exceptional intelligence, and there is no centralized decision-making process.

Swarm Behavior

One of the central tenets gleaned from swarm research has been the notion of emergent behavior. When a number of individuals are given simple rules for complex behavior, some behaviors seem to arise despite there being no rule or instruction to create them. Consider the artificial life program created by Craig Reynolds in 1986, which simulates bird flocking. Each individual bird was given a simple set of rules:

- Avoid crowding local flockmates (separation).
- Steer towards the heading average of local flockmates (alignment).
- Steer to travel in the direction of the average center of local flockmates (cohesion).

When he let the birds free, his experimental birds behaved like a real bird flock. He discovered he could add more rules to make more complex flocking behavior like goal seeking or obstacle avoidance.

Applications of Swarm Intelligence

Swarm Intelligence can be applied in many areas. Military applications include research into techniques for unmanned vehicle control. NASA is considering swarm tech for mapping planets. In a 1992 paper, George Bekey and M. Anthony Lewis discussed using swarm intelligence in nanobots inserted into the human body to attack and destroy cancer tumors.

Ant-based routing

In the telecommunication industry, the use of Swarm Intelligence has been researched using a routing table where small control packets (ants) are rewarded for successfully traversing a route. Variations on this research include forwards, backward, and bi-directional rewards. Such systems are not repeatable because they behave randomly, so commercial uses have thus far proved elusive.

One promising application for Swarm Intelligence is wireless communication networks. In this case, the network relies on a limited number of locations that are expected to provide adequate coverage for users. In this case, the application of a different kind of ant-based swarm intelligence, stochastic diffusion search (SDS) has modeled this problem with great success.

Airlines have experimented with ant-based Swarm Intelligence. Southwest Airlines uses software that employs swarm theory to manage its airlines on the ground. Each pilot acts like an “ant” in the swarm, discovering through experience what gate is best for him or her. This behavior turns out to be the best for the airline as well. The pilot colony uses the gates each one is familiar with and so can arrive at and leave from quickly, with the software providing feedback should a particular gate or route be predicted to suffer a back-up.

Crowd Simulation

The movies are using Swarm Intelligence simulations for depicting animal and human crowds. In *Batman Returns*, Swarm Intelligence was employed to create a realistic bat simulation. For *The Lord of the Rings* movies, Swarm

Intelligence simulations were employed to depict the massive battle scenes.

Human Swarms

When combined with mediating software, a network of distributed people can be organized into swarms by implementing closed-loop, real-time control systems. These systems, acting out in real-time, allow human actors to act in a unified manner, a collective intelligence that operates like a single mind, making predictions or answering questions. Testing in academic settings suggests these human swarms can out-perform individuals in a wide variety of real-world situations.

Swarm Intelligence and Machine Intelligence are both forms of artificial intelligence. It's debatable whether Swarm Intelligence is a sub-set of Machine Intelligence. It is a different approach toward the goal of smart machines, modeling the behavior of particular kinds of animals to achieve desired results. But however they are classified, Swarm Intelligence and Machine Intelligence can complement each other. In an attempt to determine emotions from a text, a Swarm Intelligence approach will differ from a monolithic approach. Instead of one Machine Learning algorithm to detect emotion in text, a swarm approach might be to create many simple Machine Learning algorithms, each designed to detect a single emotion. These heuristics can be layered in hierarchies to avoid any emotion-detector fouling up the end result. Let's look at an example. Imagine a Machine Learning swarm meant to detect emotion in written text examining this sentence: "When I pulled back the covers of my bed this morning, a giant spider ran over my leg, and I ran out of the bedroom screaming."

This is a complex sentence and very difficult for natural language Machine Learning algorithms to parse for emotion. However, a swarm of simple Machine Learning algorithms dedicated to detecting one kind of emotion would likely have the terror algorithm scoring high, while fun and happiness scored low.

So far our imaginary system is working well. But take a more difficult example:

"I watched the game yesterday and saw us kick butt in the first period, but by the third, I was terrified we would lose."

Human beings understand hyperbole. The writer is not terrified but anxious the team will lose the game. Our swarm Machine Intelligence algorithms could have the fear/terror algorithm scoring high, but this would be inaccurate. Because swarm models can be hierarchical, one model's output could be the input of another. In this case, a master model that detects emotion could filter through the outputs of each individual emotion algorithm, noting that "excitement" had been triggered by "kick butt," parse that the subject of the sentence is a sport, and determine that anxiety is a better fit than terror.

It seems fair to define Swarm Intelligence as an application of Machine Learning with an extremely narrow focus.

Chapter 16: Machine Learning Models

There are many models of Machine Learning. These theoretical models describe the heuristics used to accomplish the ideal, allowing the machines to learn on their own.

Decision Tree

Just about everyone has used the decision tree technique. Either formally or informally, we decide on a single course of action from many possibilities based on previous experience. The possibilities look like branches, and we take one of them and reject the others.

The decision tree model gets its name from the shape created when its decision processes are drawn out graphically. A decision tree offers a great deal of flexibility in terms of what input values it can receive. Also, a tree's outputs can take the form of a category, binary, or numerical. The strength of decision trees is how the degree of influence of different input variables can be determined by the level of decision node in which they are considered.

A weakness of decision trees is the fact that every decision boundary is a forced binary split. There is no nuance. Each decision is either yes or no, one or zero. Moreover, the decision criteria can consider only a single variable at a time. There cannot be a combination of more than one input variable.

Decision trees cannot be updated incrementally. That is to say, once a tree has been trained on a training set, it must be thrown out and a new one created to tackle new training data.

Ensemble Methods address many tree limitations. In essence, the ensemble method uses more than one tree to increase output accuracy. There are two main ensemble methods — bagging and boosting.

The bagging ensemble method (known as Bootstrap Aggregation) is meant to reduce decision tree variance. The training data is broken up randomly into subsets, and each subset is used to train a decision tree. The results from all trees are averaged, providing a more robust predictive accuracy than any single tree on its own.

The boosting ensemble method resembles a multi-stage rocket. The main booster of a rocket supplies the vehicle with a large amount of inertia. When its fuel is spent, it detaches, and the second stage combines its acceleration to the inertia already imparted to the rocket and so on. For decision trees, the first tree operates on the training data and produces its output. The next tree uses the earlier tree's output as its input. When the input is in error, the weighting it is given makes it more likely the next tree will identify and at least partially mitigate this error. The end result of the run is a strong learner emerging from a series of weaker learners.

Linear Regression

The premise of linear regression methods rests on the assumption that the output (numeric value) may be expressed as a combination of the input variable set (also numeric). A simple example might look like this:

$$x = a_1y_1, a_2y_2, a_3y_3...$$

Where x is the output, $a_1...a_n$ are the weights accorded to each input, and $y_1...y_n$ are the inputs.

A weakness of the linear regression model is the fact that it assumes linear input features, which might not be the case. Inputs must be tested mathematically for linearity.

K-Means Clustering Algorithm

K-means is an unsupervised machine learning algorithm for cluster analysis. It is an iterative, non-deterministic method. The algorithm operates on data sets using predefined clusters. You can think of clusters like categories. For example, consider K-Means Clustering on a set of search results. The search term "jaguar" returns all pages containing the word Jaguar. But the word "Jaguar" has more than one classification. It can refer to a type of car, it can refer to an operating system created by the Apple Computers, and it can refer to the animal. K-Means clustering algorithms can be used to group results that talk about similar concepts^[8]. So, the algorithm will group all results that discuss jaguar as an animal into one cluster, discussions of Jaguar as a car into another cluster, and discussions of Jaguar as an operating system into a third. And so on.

K-Means Clustering Applications

K-Means Clustering algorithms are used by most web search engines to cluster web pages by similarity and to identify the relevance of search results. But in any application where unstructured data needs to be divided into meaningful categories, K-Means Clustering is a valuable tool to accomplish this task.

Neural Network

We have covered neural networks in detail above. The strengths of neural networks are their ability to learn non-linear relationships between inputs and outputs.

Bayesian Network

Bayesian networks produce probabilistic relationships between outputs and inputs. This type of network requires all data to be binary. The strengths of the Bayesian network include high scalability and support for incremental learning. We discussed Bayesian models in more detail earlier in the book. In particular, this Machine Learning method is particularly good at classification tasks such as detecting if an email is or is not spam.

Support Vector Machine

Support Vector Machine algorithms are supervised Machine Learning algorithms that can be used for classification and regression analysis, although they are most often used for classification problems, and so, we will focus on those.

Support Vector Machines work by dividing categories using a hyperplane (basically a flat sheet in 3 dimensions). On one side of the hyperplane will be one category, on the other side the other category. This works well if the categories to be separated are clearly divisible, but what if they overlap? Imagine a series of blue and red dots spread out randomly within a square. There is nowhere to place a hyperplane between the blue and red dots because they overlap each other on the two-dimensional sheet. Support Vector Machines deal with this problem by mapping the data into a higher dimension. Imagine these dots being lifted into the air in a cube, where the red dots are “heavier” than the blue dots, and therefore remain closer to the

bottom and the blue dots closer to the top. If there is still an overlap, the Support Vector Machine algorithm maps them into yet another dimension, until a clear separation appears and the two categories of dots can be separated by the hyperplane.

This is an incredibly powerful method for classifying data, but there are issues, of course. One is the fact that looking at the data once it has been mapped into higher dimensions isn't possible. It has become gibberish. Support Vector Machines are sometimes referred to as black boxes for this reason. Also, Support Vector Machine training times can be high, so this algorithm isn't as well suited to very large data sets, as the larger the data set, the longer the training required.

Chapter 17: Applications of Machine Learning

You are probably using real-world applications of Machine Learning every day. Some are easily recognized, others are hidden and operating in the background, making our lives easier, or more convenient, or even safer. Though you might not recognize it, there are many technological advancements in your home that work through programming and training to become more accessible. Your devices' abilities to learn to provide the basis for modern technology.

Virtual Personal Assistants

Google Assistant, Apple's Siri, Amazon's Alexa, and Microsoft's Cortana – these are just a few of the virtual assistants we interact with almost every day. These are probably the most popular and well-known examples of applied Machine Learning. When you use these personal assistants, there is more than one kind of Machine Learning working behind the scenes. First, you can speak to them using ordinary language, and they answer back the same way. Speech recognition is a Machine Learning skill these assistants use to understand you. Then, of course, once they have recognized what you said and understood what you are asking for, they usually use yet more Machine Learning to search for answers to your questions. Finally, they respond with answers and track your response, compare it to previous responses, and use all this data to be more accurate and efficient for your future requests.

Commuting Predictions

If you've used Google or Apple maps, you've used Machine Learning to get yourself to your destination. These apps store your location, direction, and speed in a central location to lead you to your destination, providing turn details and so on before you actually reach the turning point. At the same time, they are aggregating the data from all of the users nearby who are using their service and using all this information to predict and track traffic congestion and to modify your route in real-time to avoid it. When there are not enough people using the app at any given time, the system relies on predictive Machine Learning from previous traffic data from different days at

the same location to anticipate the traffic if it is unable to collect in real-time.

Uber and Lyft also rely on Machine Learning to decide on the cost of a ride, basing their decisions on current demand, as well as predictions concerning rider demand at any given time. These services would become technically impossible without Machine Learning crunching massive amounts of data, both historical and in real-time.

Online Fraud Detection

Securing cyberspace is one of the many goals of Machine Learning. Tracking online fraud is an important tool for achieving this goal. PayPal employs Machine Learning in its efforts to prevent money laundering. Their software can track the millions of transactions taking place and separate legitimate from illegitimate transactions going on between buyers and sellers.

Data Security

Another form of security in cyberspace is combating malware. This is a massive problem and getting worse. 1 in 13 web requests led to malware on the internet in 2018, and that's up 3% from 2016. The problem is so vast only something as powerful as Machine Learning is able to cope. Fortunately, most new malware is really just older malware with minor changes. This is something Machine Learning can search for quickly and with a high degree of accuracy, meaning new malware is caught almost as quickly as it can be created.

Machine Learning is also very good at monitoring data access and finding anomalies, which might predict possible security breaches.

Personal Security

Do you face long lines at the airport? Does it take forever to get into that concert? Those long lines are for your security. Everyone getting on a plane or entering a large event is screened to ensure the safety of the flight or event. Machine Learning is starting to provide assistance to the manual checking of people, spotting anomalies human screenings might miss, and helping to prevent false alarms. This help promises to speed up security screening in a substantial way, while at the same time ensuring safer events through the more powerful screening processes Machine Learning can provide.

Video Surveillance

How many movies have you watched where a video feed is interrupted with a loop of a parade or cell, so the good guys (or bad guys) can move through the monitored area without being seen? Well, Machine Learning may not be able to defeat this tactic any better than the hapless security guards in the movies, but it is able to take over a lot of the drudgery of monitoring live video feeds. The benefit is Machine Learning never gets tired, or needs a break, or has its attention wander, so nothing is ever missed. Machine learning can focus on anomalous behavior like standing for a long time without moving, sleep on benches, or stumbling, meaning human beings can be freed up to do what they do much better, deal with these people as required.

Social Media Services

Social media platforms utilize many forms of Machine Learning. If you've been using these platforms, then you've been using Machine Learning. Facebook, for example, constantly offers you friend suggestions of "people you may know." This is a simple concept of learning through experience. Facebook's technology watches who your friends are (and who those friend's friends are), what profiles you visit, and how often, what articles you follow, and pages you visit. By aggregating this information, Machine Learning is able to provide people you are likely to enjoy interacting with and so recommends them to you as Friend suggestions.

Or when you upload a picture to Facebook of yourself and some friends, Facebook uses Machine Learning to identify those people in your photo by comparing them to images your friends have uploaded of themselves. But this is not all. These photos are scanned for the poses people are making, as well as any unique features in the background, Geo-locating the image if it can.

Online Customer Support

Many websites offer a service to text chat with customer service while you browse their pages. You've probably chatted with one or more of them. But what you might not know is, not all of those text chats are with real people. Many companies cannot afford to pay a person to provide this service, so instead a chatbot will answer your questions, itself relying on details from the

website to provide its answers. Machine Learning means these chatbots get better over time, learning how and how not to answer questions, and to seem more human in their interactions.

Search Engine Results, Refined

Search engine providers rely on Machine Learning to refine their responses to search terms. They monitor your activity after you've been given some results to examine. Do you click on the first link and stay on this page for a long time? Then their results to your query were probably good. Or do you navigate to page 2 or 3 or further down the results they have provided, clicking some links and immediately returning? This means the results returned did not satisfy your query as well as they could. Machine Learning uses this feedback to improve the search algorithms used.

Product Recommendations/Market Personalization

Have you ever been shopping online for something you didn't intend to buy? Or were you considering buying it, but not for a while? So you look online at a product, maybe at the same product in more than one online store, comparing prices, customer reviews, and so on. You might even put one of them in your shopping cart to checkout taxes and shipping fees. Then, you close these tabs and get back to work.

Over the next few days, you realize you see this product and similar products all over web pages you visit. Advertisements for the product seem to be everywhere for the next few days.

Using the kind of information gathered from your shopping/browsing practices, Machine Learning is used to tailor everything from specific email campaigns aimed at you personally, to similar product advertisements, to coupons or deals by the retailers offering the product you'd been looking at. As the amount of data for these Machine Learning algorithms grows, expect this personal approach to advertising to become even more focused and accurate.

Financial Trading

Being able to "predict the market" is something of a holy grail in financial circles. The truth is humans have never been much better at this than chance.

However, Machine Learning is bringing the grail within the human grasp. Using the power of massive supercomputers crunching enormous quantities of data, large financial firms with proprietary predictive systems make a large volume of high-speed transactions. With high speed and enough volume, even low-probability trades can end up making these firms enormous amounts of money. What about those of us who can't afford massive supercomputers? We're better off doing our stock trading the old-fashioned way — research specific companies and invest in businesses with solid business plans and room in the market for growth. No doubt one day soon, there will be a Machine Learning algorithm to manage this for you.

Healthcare

Machine Learning is revolutionizing the health care system. With its ability to ingest enormous amounts of data and use that to spot patterns human simply cannot see, Machine Learning is able to diagnose some diseases up to a year before a human diagnosis. At the same time, by crunching large data sets of populations, Machine Learning is able to identify groups or individuals who are more likely to need hospitalization due to diseases like diabetes. The predictive power of Machine Learning is likely the most fruitful avenue in healthcare since much of disease-fighting rests on early diagnosis.

Another inroad to healthcare by Machine Learning is robotic assistants in the operating room. While many of these robots are simply tools used by doctors, some are semi-autonomous and aid in the surgery themselves. In the not too distant future, Machine Learning will allow robot surgeons to perform operations with complete autonomy. This will free up surgeons to perform the far more complex and experimental procedures that Machine Learning is not prepared to perform.

Natural Language Processing

Being able to understand natural human speech has been anticipated for a long time. We've been talking to our robots in movies and cartoons for more than 70 years. And now, thanks to Machine Learning, that ability is being realized all around us. We can talk to our smartphones, and we can talk to a company's "customer support" robot and tell it what we need help with. Natural language processing is even able to take complex legal contract

language and translate it into plain language for non-lawyers to understand.

Smart Cars

The autonomous car is very close. But it won't just be autonomous; it will be smart. It will interconnect with other cars and other internet-enabled devices, as well as learn about its owner and passengers, providing internal adjustments to the temperature, audio, seat settings, and so on. Because autonomous cars will communicate with each other as they become more numerous on the road, they will become safer. Accidents will drop to next to zero when people are no longer behind the wheel. This is the power of Machine Learning really flexing its muscles, and only one of the many, many ways it will change our human future.

Chapter 18: Programming and (Free) Datasets

For those interested in the world of programming in Machine Learning, there are many, many resources on the web. Programming for small devices, like Arduino, are also available. Learning programming on these machines provides easier adaptation to other devices. This chapter will identify and describe some excellent free data sets available for download and use in Machine Learning programs, as well as suggest an option for learning beginning level Machine Learning software development using the Python programming language.

Limitations of Machine Learning

There are many problems and limitations with Machine Learning. This chapter will go over the technical issues that are currently or may in the future limit the development of Machine Learning. Finally, it will end with some philosophical concerns about possible issues Machine Learning may bring about in the future.

Concerns about Machine Learning limitations have been summarized in a simple phrase, which outlines the main objects to Machine Learning. It suggests Machine Learning is greedy, brittle, opaque and shallow. Let's examine each one in turn.

Greedy

By calling it greedy, critics of Machine Learning point to the need for massive amounts of training data to be available in order to successfully train Machine Learning systems to acceptable levels of error. Because Machine Learning systems are trained not programmed, their usefulness will be directly proportional to the amount (and quality) of the data sets used to train them.

Related to the size of training data required is the fact that, for supervised and semi-supervised Machine Learning training, the raw data used for training must first be labeled so that it can be meaningfully employed by the software to train. In essence, the task of labeling training data means to clean up the raw content and prepare it for the software to ingest. But labeling data can itself be a very complex task, as well as often a laborious and tedious one. None or improperly labeled data fed into a supervised Machine Learning system will produce nothing of value. It will just be a waste of time.

Brittle

To say Machine Learning is brittle is to highlight a very real and difficult problem in Artificial Intelligence. The problem is, even after a Machine Learning system has been trained to provide extremely accurate and valuable predictions on data it has been trained to deal with, asking that trained system

to examine a data set even slightly different from the type it was trained on will often cause a complete failure of the system to produce any predictive value. That is to say, Machine Learning systems are unable to contextualize what they have learned and apply it to even extremely similar circumstances to those on which they have been trained. At the same time, attempting to train an already trained Machine Learning algorithm with a different data set will cause the system to “forget” its previous learning, losing, in turn, all the time and effort put in to preparing the previous data sets and training the algorithm with it.

Bias in Machine Learning systems is another example of how Machine Learning systems can be brittle. In fact, there are several different kinds of bias that threaten Artificial Intelligence. Here are a few:

Bias in Data:

Machine Learning is, for the foreseeable future at least, at the mercy of the data used to train it. If this data is biased in any way, whether deliberately or by accident, those biases hidden within it may be passed onto the Machine Learning system itself. If not caught during the training stage, this bias can taint the work of the system when it is out in the real world doing what it was designed to do. Facial recognition provides a good example, where facial recognition systems trained in predominantly white environments with predominantly white samples of faces, have trouble recognizing people with non-white facial pigmentation.

Acquired Bias:

It is sometimes the case that, while interacting with people in the real world, Machine Learning systems can acquire the biases of the people they are exposed to. A real-world example was Microsoft’s Tay, a chatbot designed to interact with people on Twitter using natural language. Within 24 hours, Tay became a pretty foul troll spouting racist and misogynist statements. Microsoft pulled the plug (and set about white-washing twitter). Many of the truly offensive things Tay said were the result of people telling it to say them (there was an option to tell Tay to repeat a phrase you sent it), but there were some very troubling comments produced by Tay that it was not instructed to repeat. Microsoft was clearly aware of how nasty Twitter can get, and I think

it's fair to say creating a racist, misogynist chatbot was just about the last thing on their whiteboard. So if a massive, wealthy company like Microsoft cannot train an artificial intelligence that doesn't jump into the racist, woman-hating camp of nasty internet trolls, what does that say about the dangers inherent in any artificial intelligence system we create to interact in the real world with real people?

Emergent Bias:

An echo chamber is a group or collection of people who all believe the same things. This could be a political meeting in a basement or a chat room on the internet. Echo chambers are not tolerant of outside ideas, especially those that disagree with or attempt to refute the group's core beliefs. Facebook has become, in many ways, the preeminent echo chamber producer on Earth. But while the echo chamber phrase is meant to describe a group of people with similar ideas, Facebook's artificial intelligence has taken this idea one step further: to an echo chamber of one. The Machine Learning system Facebook deploys to gather newsfeeds and other interesting bits of information to show to you can become just such an echo chamber. As the artificial intelligence learns about your likes and dislikes, about your interests and activities, it begins to create a bubble around you that, while it might seem comforting, will not allow opposing or offensive views to reach you. The area where this is most alarming is news and information. Being constantly surrounded by people who agree with you, reading the news that only confirms your beliefs, is not necessarily a good thing. How we know we are correct about a particular issue is by testing our beliefs against who believe otherwise. Do our arguments hold up to their criticism or might we be wrong? In a Facebook echo chamber of one, that kind of learning and growth becomes less and less possible. Some studies suggest that spending time with people who agree with you tends to polarize groups, making the divisions between them worse.

Goals Conflict Bias:

Machine Learning can often support and reinforce biases that exist in the real world because doing so increases their reward system (a system is "rewarded" when it achieves a goal). For example, imagine you run a college, and you want to increase enrollment. The contract you have with the

advertising company is that you will pay a certain amount for each click you get, meaning someone has seen your advertisement for the college and was interested enough to at least click on the link to see what you are offering. Of course, a simple generic ad like “Go to College!” wouldn’t work so well, so you run several ads simultaneously, offering degrees in engineering, teaching, mathematics, and nursing.

It is in the advertiser’s best interest to get as many clicks to your landing pages as possible. So they employ a Machine Learning algorithm to track who clicks on what advertisement, and it begins to target those groups with the specific ads to gain more clicks. So far, from the outside, this seems like a win-win. The advertiser is making more revenue, and your college is receiving more potential students examining your courses. But then you notice something in the aggregate data of link clicks. Most of the clicks you are receiving for math and engineering are from young men, while most of the clicks for nursing and teaching are from young women. This aligns with an unfortunate cultural stereotype that still exists in western culture, and your college is perpetuating it. In its desire to be rewarded, the Machine Learning assigned maximizing clicks for your advertising campaign found an existing social bias and exploited it to increase revenue for its owner. These two goals, increasing your enrollment and reducing gender bias in employment, have come into conflict and Machine Learning sacrificed one to achieve the other.

Opaque

One of the main criticisms of Machine Learning, and in particular, against Neural Networks, is that they are unable to explain to their creators why they arrive at the decisions they do. This is a problem for two reasons: one, more and more countries are adopting internet laws that include a right to an explanation. The most influential of these laws is the GDPR (The EU General Data Protection Regulation), which guarantees EU citizens the right to an explanation why an algorithm that deals with an important part of their lives made the decision it did. For example, an EU citizen turned down for a loan by a Machine Learning algorithm has a right to demand why this happened. Because some artificial intelligence tools like neural networks are often not capable of providing any explanation for their decision, and the fact this

decision is hidden in layers of math not readily available for human examination, such an explanation may not be possible. This has, in fact, slowed down the adoption of artificial intelligence in some areas. The second reason it is important that artificial intelligence be able to explain its decisions to its creators is for verifying that the underlying process is in fact meeting expectations in the real world. For Machine Learning, the decision-making process is mathematical and probabilistic. In the real world, decisions often need to be confirmed by the reasoning used to achieve them.

Take, for example, a self-driving car involved in an accident. Assuming the hardware of the car is not completely destroyed, experts will want to know why the car took the actions it did. Perhaps there is a court case, and the decision of liability rests on how and why the car took the actions it did. Without transparency around the decision-making process of the software, justice might not be served in the courts, software engineers may not be able to find, and correct flaws and more people might be in danger from the same software running in different cars.

The Philosophical Objections: Jobs, Evil, and Taking Over the World

Jobs:

One of the main concerns surrounding artificial intelligence is the way these systems are encroaching upon human employment. Automation is not a new problem, and vast numbers of jobs have already been lost in manufacturing and other such industries to robots and automated systems. Because of this, some argue that this concern that machines and artificial intelligence will take over so many jobs there will be no more economy is not really a threat. We've encountered such takeovers before, but the economy shifted, and people found other jobs. In fact, some argue that the net number of jobs has increased since the loss of so many jobs to automation. So what's the problem?

The problem is this next round of job losses will be unprecedented. Artificial Intelligence will not only replace drudgery and danger. It will keep going. There are 3.5 million of them currently. How long until all trucking is handled by Machine Learning systems running self-driving trucks? And what about ride-sharing services like Uber and Lyft?

The question to ask is not what jobs will be replaced, but what jobs can't be replaced. In the Machine Learning round of employment disruption, white collar jobs are just as much in jeopardy as blue. What about accountants, financial advisers, copywriters, and advertisers?

The pro-artificial intelligence camp argues that this disruption will open up new kinds of jobs, things we can't even imagine. Each major disruption of the economy in the past that displaced many forms of employment with automation often caused the creation of new employment unforeseen before the disruption. The difference with artificial intelligence is there is no reason to believe these new forms of employment won't quickly be taken over by Machine Learning systems as well.

And all the above assumes the use of the current type of very specific

Machine Learning. What happens if researchers are able to generalize learning in artificial intelligence? What if these systems become able to generalize what they learn and apply what they know in new and different contexts? Such a generalized artificial intelligence system could very quickly learn to do just about anything.

Evil:

A very serious danger from artificial intelligence is the fact that anyone with resources can acquire and use it. Western governments are already toying with the idea of autonomous weapons platforms using artificial intelligence to engage and defeat an enemy, with little or no human oversight. As frightening as this might be, in these countries at least, there are checks and balances on the development and deployment of such devices, and in the end, the population can vote for or against such things.

But even if these platforms are developed and only used appropriately, what is to stop an enemy from capturing one, reverse engineering it, and then developing their own? What's to stop a rival power from investing in the infrastructure to create their own?

The threat of Machine Learning used by bad actors is very real. How do we control who gains access to this powerful technology? The genie is out of the bottle. It can't be put back in. So, how do we keep it from falling into the wrong hands?

Taking Over the World:

Finally, we'll take a look at the end humanity might cause by creating super-intelligent machines. Luminaries such as Steven Hawking and Elon Musk have quite publicly voiced their concerns about the dangers artificial intelligence could pose to humanity. In the Terminator movie franchise, a Defense Department artificial intelligence called Skynet, tasked with protecting the United States, determined that human beings, everywhere, were the real threat. Using its control of the US nuclear arsenal, it launched an attack on Russia, which precipitated a global nuclear war. In the disaster that followed, it began systematically wiping out the human race.

This example is extreme and probably better as a movie plot than something we have to worry about from artificial intelligence in the real world. No, the

danger to humanity from artificial intelligence lies more likely in a paperclip factory.

Imagine an automated paperclip factory run by an artificial intelligence system capable of learning over time. This system is smarter than human beings. It has one goal to produce as many paperclips as possible, as efficiently as possible. This sounds harmless enough. And at first, it is. The system learns everything there is to know about creating a paperclip, from the mining of metals, smelting, transportation of raw steel to its factory, the automated factory floor, clip design specifications, and so on.

As it learns over time, however, it runs up against walls it cannot surpass. A factory can only be so efficient. Eventually boosting efficiency further becomes impossible. Acquiring the supply chain from mining to smelting to transportation might come next. But again, once these aspects of the business are acquired and streamlined, they no longer offer a path to the goal — increase production and efficiency.

As artificial intelligence collects more information about the world, it comes to learn how to take over other factories, retool them for paperclip production, and increase output. Then more transportation, mining, and smelting might need to be acquired. After a few more factories are retrofitted, the distribution center where the paperclips are delivered begins to fill with paperclips. There are many more than anyone needs. But this is not the concern of our artificial intelligence. The supply side of the paperclip chain is not part of its programming.

It learns about politics and so begins to influence elections when people try to stop paperclip production. It continues to acquire and take over businesses and technologies to increase production.

Absurd as it sounds (and this is just a thought experiment), imagine the entire Earth, now devoid of people, a massive mining, transportation, smelting, and paperclip production facility. And as the piles of unwanted, unused paperclips turn into mountains to rival the stone ones of our planet, with metal resources dwindling, our stalwart AI turns its lenses up to see the moon, Mars, and the asteroid belt. It sees other solar systems, other galaxies, and finally the universe. So much raw metal out there, all just waiting to be

made into paperclips...

Chapter 19: Machine Learning and the Future

After the previous chapter, it's probably best to end on a positive note, by an examination of the positive future of Machine Learning. This chapter will break down the future of Machine Learning into segments of society and the economy in an effort to put these possible futures in a useful context.



Security

Facial recognition and aberrant behavior detection, these are the toolkits from Machine Learning that are available today. They will become ubiquitous in the future, protecting the public from criminal behavior and getting people in trouble the help they need.

But what about the other Machine Learning security features in the future?

In the cyber world, Machine Learning will grow and increase its influence in identifying cyber-attacks, malicious software code, and unexpected

communication attempts. At the same time, dark hat software crackers are also working on Machine Learning tools to aid them in breaking into networks, accessing private data, and causing service disruptions. The future of the cyber world will be an ongoing war between White and Black Hat Machine Learning tools. Who will win? We hope the White Hat Machine Learning algorithms will be victorious, but win or lose, the battle will slowly move out of the hands of people and into the algorithms of Machine Learning.

Another sweeping change to security we might see in the near future is autonomous drones controlled by Machine Learning algorithms. Drones can maintain constant aerial surveillance over entire cities at very little cost. With advancements in image recognition, motion capture, video recognition, and natural language processing, they will be able to communicate with people on the street, respond to natural disasters and automobile accidents, ferry medications needed in emergency situations where traditional service vehicles cannot access, and find and rescue lost hikers by leading them to safety, deliver them needed supplies, and alerting authorities of their GPS location.

Markets

The rise of Machine Learning will generate completely new Artificial Intelligence-based products and services. Entire industries will be created to service this new software, as well as new products to be added to the Internet of Things, including a new generation of robots complete with learning algorithms and the ability to see, hear, and communicate with us using natural language.

Retail

In the retail sector, we will see enhanced and more accurate personalization. But instead of merely showing us what we want, Machine Learning will be dedicated to showing us what we need. Perhaps we've been eating too much fast food this week. A smart algorithm looking out for our well-being would not throw more and more fast food ads in our face. Instead, reminders about our health, coupons for gym memberships, or recipes for our favorite salads might become part of the Artificial Intelligence toolkit for our notifications.

Machine Learning will help us to balance our lives in many ways by using knowledge about general health, and our own medical records, to provide information about not only what we want, but also what we might need.

Healthcare

Machine Learning will know almost everything about us and not only through records on the internet. When we visit our doctor to complain about a sore shoulder, Machine Learning might inform our GP about how we are prone to slouch at work, possibly altering the doctor's diagnosis from a prescription for analgesics to a prescription for exercises to do at work, as well as some tutoring on better sitting posture.

On the diagnostic side, Machine Learning will do the busy work it is best at: examining our x-rays and blood-work and mammograms, looking for patterns human beings cannot see, getting pre-emptive diagnostic information to our doctors so we can head off serious illness at early stages, or perhaps before it even happens. Doctors will be freed up to spend time with their patients, providing the human touch so often missing in medicine.

Many if not most surgeries will be performed by Artificial Intelligence-enabled robots, either assisting human surgeons, being supervised by them, or even, eventually, working fully autonomously. Machine Learning will monitor our blood gasses under anesthesia, our heart rate, and other health measures during operations, and reaction in milliseconds should something appear to be wrong. Iatrogenic disease will decrease dramatically, if not disappear completely.

The Environment and Sustainability

Machine Learning will be able to study the movement of people in cities, what they use and don't use, their patterns of use, how they travel, and where. Deep learning from this data will allow city planners to employ Machine Learning algorithms to design and construct both more efficient and pleasant cities. It will allow massive increases in density without sacrificing quality of life. These efficiencies will reduce or even possibly eliminate net carbon emissions from cities.

Augmented Reality

When we wear Google (or Microsoft or Apple or Facebook) glasses in the future, the embedded processes, video capture, audio capture, and microphones on these devices will do much more than give us directions to find a location. Machine Learning will be able to see what we see and provide explanations and predictive guidance throughout our day. Imagine having your day “painted” with relevant information on interior walls and doors, and outside on buildings and signs, guiding you through your schedule with the information you need right when you need it.

Information Technology

Machine Learning will become a tool people and businesses can apply as needed like SasS. This MLaaS will allow software to be aware of its surroundings, to see us, to hear us, and speak to us in natural language. Connected to the internet, every device will become smart, and generate an ecosphere around us that attends to our needs and concerns, often before we even realize we have them. These “Cognitive Services” will provide APIs and SDKs, leading to rapid “smart” software development and deployment.

Specialized hardware will increase the speed of Machine Learning training, as well as increase its speed in servicing us. Dedicated AI chips will bring about a huge change in Artificial Intelligence speed and ubiquity.

Microcomputers will come equipped with Machine Learning capabilities so that even the smallest device will be able to understand its surroundings. Where there is no convenient power supply, these devices will run on dime-sized batteries, lasting for months or years of service before needing replacement. Currently, these microcomputers cost about 50 cents each. This price will drop. They will be deployed practically everywhere.

Quantum computing and Machine Learning will merge, bringing solutions to problems we don’t even know we have yet.

We will see the rise of intelligent robots of every size, make and description, all dedicated to making our lives better.

Trust Barriers

Natural speech means we will be able to talk to our devices and be understood by them. The current trust barriers between some people, business

sectors, and governments will slowly break down as Machine Learning demonstrates its reliability and effectiveness. Improved unsupervised learning will reduce the time required to develop new Machine Learning software with required specifications.

Conclusion

Thank you for making it through to the end of Machine Learning for Beginners. Let's hope it was informative and able to provide you with all of the information you needed to begin to understand this extensive topic.

The impact of Machine Learning on our world is already ubiquitous. Our cars, our phones, our houses, and so much more are already being controlled and maintained through rudimentary Machine Learning systems. But in the future, Machine Learning will radically change the world. Some of those changes are easy to predict. In the next decade or two, people will no longer drive cars. Instead, automated cars will drive people. But in many other ways, the effect of Machine Learning on our world is difficult to predict. Will Machine Learning algorithms replace so many jobs, from trucking to accounting to many other disciplines, that there won't be much work left for people? In 100 years, will there be work for anyone at all? We don't know the answer to questions like these because there is so far no limit to what Machine Learning can accomplish, given time and data and the will to use it to achieve a particular task.

The future is not necessarily frightening. If there is no work in the future, it won't mean that things aren't getting done. Food will still be grown, picked, transported to market, and displayed in stores. It's just that people won't have to do any of that labor. As a matter of fact, stores won't be necessary either, since the food we order can be delivered directly to our homes. What will the world be like if human beings have almost unlimited leisure time? Is this a possible future?

The only real certainty about artificial intelligence and Machine Learning is that it is increasing in both speed of deployment and in areas which it can influence. It promises many benefits and many radical changes in our society. Learning about this technology and creating your own programming in

relation to the development of your own machinery will improve knowledge of this increasingly popular technology and skyrocket you to the head of future developments.

Conclusion

Machine learning is a branch of artificial intelligence that involves the design and development of systems capable of showing an improvement in performance based on their previous experiences. This means that, when reacting to the same situation, a machine should show improvement from time to time. With machine learning, software systems are able to predict accurately without having to be programmed explicitly. The goal of machine learning is to build algorithms which can receive input data then use statistical analysis so as to predict the output value in an acceptable range.

Machine learning originated from pattern recognition and the theory that computers are able to learn without the need for programming them to perform tasks. Researchers in the field of artificial intelligence wanted to determine whether computers are able to learn from data. Machine learning is an iterative approach, and this is why models are able to adapt as they are being exposed to new data. Models learn from their previous computations so as to give repeatable, reliable results and decisions.

References

- Agile Actors. (2019). Scikit-learn Tutorial: Machine Learning in Python - Agile Actors #learning. Retrieved from <https://learningactors.com/scikit-learn-tutorial-machine-learning-in-python/>
- Bose, D. (2019). Benefits of Artificial Intelligence to Web Development. Retrieved from <http://www.urbanui.com/artificial-intelligence-web-development/>
- Internet of things. (2019). Retrieved from https://en.wikipedia.org/wiki/Internet_of_things
- DataMafia. (2019). DataMafia. Retrieved from <https://datamafia2.wordpress.com/author/datamafia007/>
- Karbhari, V. (2019). Top AI Interview Questions & Answers — Acing the AI Interview. Retrieved from <https://medium.com/acing-ai/top-ai-interview-questions-answers-acing-the-ai-interview-61bf52ca34d4>
- Newell, G. (2019). How to Decompress Files With the "gz" Extension. Retrieved from <https://www.lifewire.com/example-uses-of-the-gunzip-command-4081346>
- Pandas, D. (2019). Different types of features to train Naive Bayes in Python Pandas. Retrieved from <https://stackoverflow.com/questions/32707914/different-types-of-features-to-train-naive-bayes-in-python-pandas/32747371#32747371>
- Robinson, S. (2019). K-Means Clustering with Scikit-Learn. Retrieved from <https://stackabuse.com/k-means-clustering-with-scikit-learn/>
- Top 10 Machine Learning Algorithms. (2019). Retrieved from <https://www.dezyre.com/article/top-10-machine-learning-algorithms/202>
- Samuel, N. (2019). Installing TensorFlow on Windows. Retrieved from <https://stackabuse.com/installing-tensorflow-on-windows/>

-
- [1] Information retrieved from Robinson, S. (2019). K-Means Clustering with Scikit-Learn. Retrieved from <https://stackabuse.com/k-means-clustering-with-scikit-learn/>
- [2] Information obtained from DataMafia. (2019). DataMafia. Retrieved from <https://datamafia2.wordpress.com/author/datamafia007/>
- [3] Information obtained from Robinson, S. (2019). K-Means Clustering with Scikit-Learn. Retrieved from <https://stackabuse.com/k-means-clustering-with-scikit-learn/>
- [4] Information obtained from Robinson, S. (2019). K-Means Clustering with Scikit-Learn. Retrieved from <https://stackabuse.com/k-means-clustering-with-scikit-learn/>
- [5] Information obtained from Pandas, D. (2019). Different types of features to train Naive Bayes in Python Pandas. Retrieved from <https://stackoverflow.com/questions/32707914/different-types-of-features-to-train-naive-bayes-in-python-pandas/32747371#32747371>
- [6] Information obtained from Pandas, D. (2019). Different types of features to train Naive Bayes in Python Pandas. Retrieved from <https://stackoverflow.com/questions/32707914/different-types-of-features-to-train-naive-bayes-in-python-pandas/32747371#32747371>
- [7] ("Internet of things", 2019)
- [8] ("Top 10 Machine Learning Algorithms", 2019)