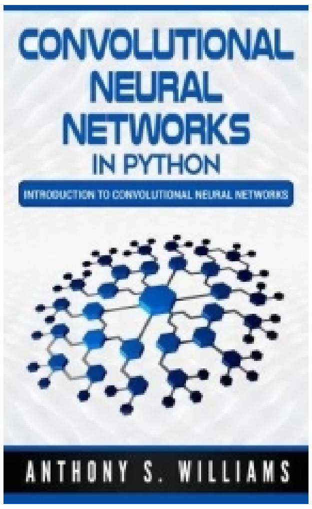
DATA ANALYTICS FOR BEGINNERS
INTRODUCTION TO DATA ANALYTICS
ANTHONY S. WILLIAMS
DEEP LEARNING WITH KERAS
INTRODUCTION TO DEEP LEARNING WITH KERAS
ANTHONY S. WILLIAMS
ANALYZING DATA WITH POWER BI
INTRODUCTION TO POWER BI
ANTHONY S. WILLIAMS
REINFORCEMENT LEARNING WITH PYTHON
A SHORT OVERVIEW OF REINFORCEMENT LEARNING WITH PYTHON
ANTHONY S. WILLIAMS
ARTIFICIAL INTELLIGENCE PYTHON
A SHORT INTRODUCTION TO ARTIFICIAL INTELLIGENCE WITH PYTHON
ANTHONY S. WILLIAMS
TEXT ANALYTICS WITH PYTHON
A BRIEF INTRODUCTION TO TEXT ANALYTICS WITH PYTHON
ANTHONY S. WILLIAMS
CONVOLUTIONAL NEURAL NETWORKS IN PYTHON
INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS
ANTHONY S. WILLIAMS


CONVOLUTIONAL NEURAL NETWORKS IN PYTHON
INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS
ANTHONY S. WILLIAMS

Convolutional Neural Networks in Python

By *Anthony S. Williams*

Data Analytics for Beginners

Introduction to Data Analytics

By Anthony S. Williams

Table of Contents

Introduction

One of the most common buzzwords floating around online today is data analysis, and while you may have heard of it, figuring out exactly what it means might be more difficult than you might expect. The reason for this is that there are several different definitions for the phrase depending on who you ask. While it can mean more specific things in context, in general, a definition that you can work with is that it is the process by which data is modeled, transformed, cleaned and inspected by businesses, with the ultimate goal being its use in the decision-making process.

As such, this makes a data analyst the person whose job it is to find the best answers to the questions that businesses come up with. They take the lines and lines of data that they find and paint a clear picture of just what it means so that those without the skills to see the pictures in the data still have a firm grasp on what is going on in the market or even with their very own businesses. The data that is analyzed varies radically based on the business that is looking and what it is they are looking for, so much so that it is currently created at a rate of more than 2 quintillion bytes each day worldwide.

This dramatic influx in what is available has not gone unnoticed, and businesses everywhere are getting in on the mega trend of collecting data and analyzing it as quickly and effectively as possible. When it comes to finding the truth at the heart of the data, the deeper and more accurate of an insight you can find, the more easily you will be able to discover the hidden trends that are hiding and use them as effectively, and profit from them.

It is especially useful as it can be used equally effectively by both automated and human-driven decisions. What it all boils down to, is that traditional business intelligence can often be used to determine what a specific problem is, how often it occurs, where an issue is located and, even, how it can be fixed, but good analytics can determine the source of the problem in the first place as well as what is likely going to happen if the current trends continue.

Evolving Usage

Analytics have been in near constant use since a variety of time-management exercises were first instituted by a man named Frederick Winslow Taylor in the 1800s. Taylor was an engineer who was fascinated with the idea of industrial efficiency and what it could mean for businesses of all shapes and sizes. While working as a foreman at the Midvale Steel Works, Taylor expected his team to work as hard as they could without causing themselves injury or undue stress. To determine how to do this, he started by analyzing the productivity of the men in his charge as well as the machines that they were working with. Taylor went on to use his insights to shape the field of scientific management, of which data analytics is a part.

While used in a limited fashion, most notably by Henry Ford to dictate the pacing of his assembly lines, the value of analytics was not truly grasped by a wider group of individuals until computers started to be able to utilize it as a way of implementing decision support systems in the 1960s. With the power of the burgeoning computer sciences behind then, analytics have since evolved into numerous different applications, software tools, and hardware, and include things as varied as data warehouses and enterprise resource-planning systems.

What this means for businesses today is that every business has some data that is crucial for their operations and extended success and existence. Knowing what this data is and utilizing it effectively are two different things, however, which is where data analytics comes in.

Thank you for downloading this book. I hope this book will provide answers to all of your questions concerning this topic.

In the meantime, I would like to ask you for a small favor. Could you please leave a review on this book before you continue reading?

Just click here to leave a review for this book on Amazon!

Thank you very much!

Chapter 1 Putting Data Analytics to Work

When it comes to analyzing the data that you are presented with, the first thing you are going to want to do is determine if the data that you are preparing to analyze is going to add real value to your business as a whole or if the costs are not going to be worth the time and effort needed to gather them properly. For example, going through all of your sales data in order to determine the most popular product or service you provide, as well as which is the most profitable, will provide you with clear pillars of your business to focus on in order to ensure you are as successful as possible in both the short and the long term.

This activity is a productive use of time because it can help you to accurately predict what the future of your business could look like under certain market conditions. Once those conditions have been properly pinpointed, they can then be used as a direction for the business to move in the short term. By default, the above example also serves another, potentially more important, purpose; it shows you what products or services that you are offering that absolutely no one is interested in taking you up on. As such, you would then be able to more accurately determine if there might not be a better use of your company's resources than the underperforming products. Either way, a byproduct of the process is a reduction of waste as well as an increase in sales revenue.

Targeting the right data: To understand the right type of data you are going to want to target, it is important to understand that it primarily comes in two forms, those that are structured, such as traditional databases, and those that are unstructured, things like social media networks or phone applications that rely on data sent over the internet. With so much data to keep track of, you may find it helpful to set up an automated method of collection when it comes to your daily transaction output to ensure that any future analysis is done with data that is as straight from the source as possible.

This, in turn, means that you will need to determine the right database structure for the data that you are going to be retrieving and the way you are going to want to access it later if you hope to see the best results. Choosing the right type of database for your business is an important choice and one which is discussed several times in the following pages.

However, if you instead plan on determining the current public sentiment that your customers have towards your business, then you are going to want to analyze online applications and social media profiles, the two places that people are prone to leaving feedback in the ever-connected world of the twenty-first century.

Determine the best type of analysis: After you have found the data that you are looking for, the next thing you are going to want to do is determine the most effective type of analysis that you can use to get at the heart of the data. There are numerous different tools as well as platforms for analysis on the market and determining the right one for you is going to mean looking at the ways your business operates and coming up with the right solution based on what you find.

Chapter 2   The Rise of Data Analytics

Ever met someone who had much knowledge but never knew what to do with it? How about someone who claims to be broke but unknowingly has possessions of great value? Have you ever seen someone who feels that they have something, so they ask another to help them identify it? Welcome to the world of Data Analysis.

The art of gathering information - that's what it is. What do you do with millions and millions of data floating around? This arena of information is what market researchers use as a door-opener to a lot of things that share one common answer - information. In marketing research, there are issues, new or old products, updates, downgrades, and then there are the providers and the consumers - basically, queries. To answer the questions, the right questions must be asked, the right crowd must be interviewed, and the right approach must be used.

The need for data increases as many seek to improve, create, achieve, satisfy and discover. As marketing research continues to expand, data analytics only has one way to go in the industry - up. Methods of gathering data have increased as accessing technology requires information in exchange. Here are some examples:

1. Phone Applications - Before downloading an app, you are always asked to sign up. Signing up leads to giving information. After a while of using the app, a pop up comes out and asks you for comments or ratings. There are also certain apps that automatically direct you to an online survey after uninstalling them.

2. For active respondents, mobile research allows constant communication with researchers through their marketing research websites or marketing research apps which are available for smartphones.

3. Social networking sites like Facebook, Twitter, Google+ are usually used to gather information on a more open platform because most times user comments and posts are meant to be shared with friends, family, or people who share common interests making information more spontaneous.

4. Other applications like Pinterest, Instagram, Flipped, YouTube, LinkedIn all encourage the provision of information from both a user and provider.

5. Crowd Sourcing is also another way to find information. Putting something out into the public and letting it roll sure has produced many different outcomes.

There are other ways to collect answers. Not just online, but even in person. But, why gather so much data? It's the treasure of marketing research. If you have a query, you can find the answer to it - or maybe another question. If there is no question, then whatever you find might come in handy anytime soon. And since marketing research companies are getting more in demand these days and the need for data grows proportionally with it as well.

Chapter 3 Big Data Defined

Due to the often fluid nature of the data analytics field, it is perfectly natural for certain myths to crop up surrounding its specifics. The biggest and most outlandish of these is that data analytics are only useful for large corporations or businesses with more data coming in than they can analyze. In reality, however, it is important to understand that analyzing the data that you do have available is an excellent choice regardless of the size of your business or the amount of data you can access.

Focusing on the size of your business or the limited nature of the data that you can find means focusing on the wrong issues. Instead, it is important that you look to determine if the data that you do have access to can actually be useful in a real and meaningful way. If you have access to data that you think can be useful then, it is important to seek out ways to utilize it, to your benefit.

These myths are often further segmented once concepts like Big Data enter the conversation. While the term ‘big data’ is new, however, the data that it represents has been around for nearly 20 years if not longer. Big Data can essentially be thought of as all of the data that is owned by a company and also what the company does with that data, which can be scrutinized for relative trends. The problem with Big Data, however, is in the name, which means that there is just so much of it to go through that it can be difficult to see the big picture without organizing it.

Chapter 4 Cluster Analysis

Cluster analysis is used for automatic identification of natural groupings of things. It is also known as the segmentation technique, this is well shown in Figure 4.1 below. In this technique, data instances that are similar to (or near) each other are categorized into one cluster. Similarly, data instances that are very different (or far away) from each other are moved into different clusters.

Clustering is an unsupervised learning technique as there is no output or dependent variable for which a right or wrong answer can be computed. The correct number of clusters or the definition of those clusters is not known ahead of time. Clustering techniques can only suggest to the user how many clusters would make sense from the characteristics of the data. The user can specify a different, larger or smaller, number of desired clusters based on their making business sense. The cluster analysis technique will then define many distinct clusters from analysis of the data, with cluster definitions for each of those clusters. However, there are good cluster definitions, depending on how closely the cluster parameters fit the data.

Applications of Cluster Analysis

Cluster analysis is used in almost every field where there is a large variety of transactions. It helps provide characterization, definition, and labels for populations. It can help identify natural groupings of customers, products, patients, and so on. It can also help identify outliers in a specific domain and thus decrease the size and complexity of problems. A prominent business application of cluster analysis is in market research. Customers are segmented into clusters based on their characteristics—wants and needs, geography, price sensitivity, and so on. Here are some examples of clustering:

1. Market Segmentation: Categorizing customers according to their similarities, for instance by their common wants and needs, and propensity to pay, can help with targeted marketing.

2. Product Portfolio: People of similar sizes can be grouped together to make small, medium and large sizes for clothing items.

3. Text Mining: Clustering can help organize a given collection of text documents according to their content similarities into clusters of related topics.

Definition of a Cluster

An operational definition of a cluster is that, given a representation of n objects, find K groups based on a measure of similarity, such that objects within the same group are alike but the objects in different groups are not alike.

However, the notion of similarity can be interpreted in many ways. Clusters can differ in terms of their shape, size, and density. Clusters are patterns, and there can be many kinds of patterns. Some clusters are the traditional types, such as data points hanging together. However, there are other clusters, such as all points representing the circumference of a circle. There may be concentric circles with points of different circles representing different clusters. The presence of noise in the data makes the detection of the clusters even more difficult.

An ideal cluster can be defined as a set of points that is compact and isolated. In reality, a cluster is a subjective entity whose significance and interpretation requires domain knowledge.

Representing Clusters

The clusters can be represented by a central or modal value. A cluster can be defined as the centroid of the collection of points belonging to it. A centroid is a measure of central tendency. It is the point from where the sum total of squared distance from all the points is the minimum. A real-life equivalent would be the city center as the point that is considered the easiest to use by all constituents of the city. Thus all cities are defined by their centers or downtown areas.

A cluster can also be represented by the most frequently occurring value in the cluster, i.e. the cluster can be defined by its modal value. Thus, a particular cluster representing a social point of view could be called the 'soccer moms', even though not all members of that cluster need currently be a mom with soccer-playing children.

Clustering Techniques

Cluster analysis is a machine-learning technique. The quality of a clustering result depends on the algorithm, the distance function, and the application. First, consider the distance function. Most cluster analysis methods use a distance measure to calculate the closeness between pairs of items. There are two major measures of distances: Euclidian distance ("as the crow flies" or straight line) is the most intuitive measure. The other popular measure is the Manhattan (rectilinear) distance, where one can go only in orthogonal directions. The Euclidian distance is the hypotenuse of a right triangle, while the Manhattan distance is the sum of the two legs of the right triangle. There are other measures of distance like Jacquard distance (to measure similarity of sets), or Edit distance (similarity of texts), and others.

In either case, the key objective of the clustering algorithm is the same:

- Inter-clusters distance maximized; and

- Intra-clusters distance minimized

There are many algorithms to produce clusters. There are top-down, hierarchical methods that start with creating a given number of best-fitting clusters. There are also bottom-up methods that begin with identifying naturally occurring clusters.

The most popular clustering algorithm is the K-means algorithm. It is a top-down, statistical technique, based on the method of minimizing the least squared distance from the center points of the clusters. Other techniques, such as neural networks, are also used for clustering. Comparing cluster algorithms is a difficult task as there is no single right number of clusters. However, the speed of the algorithm and its versatility in terms of different dataset are important criteria.

1. Here is the generic pseudo code for clustering
2. Pick an arbitrary number of groups/segments to be created
3. Start with some initial randomly-chosen center values for groups
4. Classify instances to closest groups
5. Compute new values for the group centers
6. Repeat step 3 & 4 till groups converge
7. If clusters are not satisfactory, go to step 1 and pick a different number of groups/segments

The clustering exercise can be continued with a different number of clusters and different location of those points. Clusters are considered good if the cluster definitions stabilize, and the stabilized definitions prove useful for the purpose at hand. Else, repeat the clustering exercise with a different number of clusters, and different starting points for group means.

Selecting the Number of Clusters

The correct choice of the value of K is often ambiguous. It depends on the shape and scale of the distribution points in a data set and the desired clustering resolution of the user. Heuristics are needed to pick the right number. One can graph the percentage of variance explained by the clusters against the number of clusters. The first clusters will add more information (explain a lot of variance), but at some point the marginal gain in variance will fall, giving a sharp angle to the graph, looking like an elbow. Beyond that elbow point, adding more clusters will not add much incremental value. That would be the desired K.

To engage with the data and to understand the clusters better, it is often better to start with a small number of clusters such as 2 or 3, depending upon the data set and the application domain. The number can be increased subsequently, as needed from an application point of view. This helps understand the data and the clusters progressively better.

Advantages and Disadvantages of K-Means algorithm

There are many advantages of K-Means Algorithm

1. K-Means algorithm is simple, easy to understand and easy to implement.
2. It is also efficient, in that the time taken to cluster K-means, rises linearly with the number of data points.
3. No other clustering algorithm performs better than K-Means, in general.

There are a few disadvantages too:

1. The user needs to specify an initial value of K.
2. The process of finding the clusters may not converge.

3. It is not suitable for discovering clusters shapes that are not hyper-ellipsoids (or hyper-spheres).

Neural networks can also be deployed for clustering, using the appropriate objective function. The neural network will produce the appropriate cluster centroids and cluster population for each cluster.

Conclusion

Cluster analysis (see example in Figure below) is a useful, unsupervised learning technique that is used in many business situations to segment the data into meaningful small groups. K-Means algorithm is an easy statistical technique to iteratively segment the data. However, there is only a heuristic technique to select the right number of clusters.

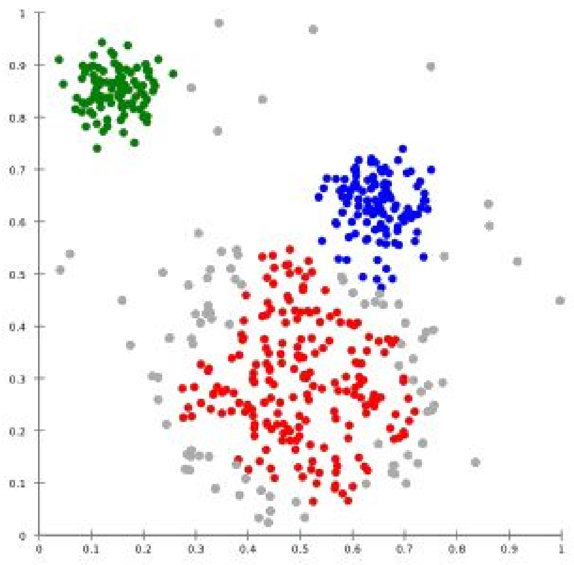


Figure: Cluster Analysis

Chapter 5 Data Mining

Data mining is the name that has been given to the process of finding the relevant trends within Big Data and then analyzing it for the benefit of the additional perspective it brings, before going on to find additional relationships hidden within the data. Data mining as a whole can be split into two primary types: descriptive, which is described in this chapter; and predictive, which is discussed in the next chapter.

Data Mining Basic Requirements

If you plan on mining data on a regular basis, then the first thing you will need to do is to ensure that your data warehouse is in order as previously discussed. Additionally, you are going to want to find data analysis tools that are easy enough to use and comprehend that you don't need someone whose entire purpose is to know how to use them. Finally, you are going to want to ensure that the information that you do generate is going to be compatible with numerous different systems.

When it comes to deciding what tools you are going to want to use, you may find it useful to determine how they are going to be used in a conventional decision-making process. The first step of the decision-making process is to develop a style of reporting that is standardized. Next, it will be important to take note of any instances that might be an exception to the rule you have created. These exceptions could be positive and lead to advantages, or they could be negative and give you an insight into potential problems. Once exceptions have been noted you will want to determine important causes before looking into alternatives and determining the overall of what it is that you have decided.

Standard reports are any results that you pull using database queries, and they can determine how a business is performing as well as shed light on several other important business factors.

When exceptions occur, however, then you want to know that the details will be easy to retrieve when needed.

Types of Data Mining Techniques

Raw data is not any different from the crude oil. Today, any person or institution with a moderate budget can collect large volumes of raw data. But the collection in itself shouldn't be the end goal. Companies that can extract meanings from the collected raw data are the ones that can compete in today's complex and unpredictable environment, this can be seen inFigure 5.1 below.

At the core of any data refinement process sits what we are referring to as "data mining techniques." Here are common data mining techniques:

- Descriptive
- Diagnostic
- Prescriptive
- Exploratory
- Predictive
- Mechanistic
- Casual
- Inferential

Let's dive in and explore these technologies.

#1: Descriptive Analytics

The primary focus of descriptive analytics is to summarize what happened in an organization. Descriptive Analytics examines the raw data or content—that is manually performed—to answer questions such as:

- What happened in the organization?
- What is happening in the organization?

Descriptive data mining analytics is characterized by conventional business intelligence and visualizations such as the bar charts, pie charts, line graphs, or the generated narratives. A simple illustration of descriptive analytics can be assessing credit risk in a bank where the past financial performances help to predict client's expected financial performance. Descriptive analytics is useful in providing insights into sales cycle such as categorizing customers based on their preferences.

#2: Diagnostic Analytics

As the name suggests, diagnostic data mining technique is used to unearth or to determine why something happened. For example, if you're conducting a social media marketing campaign, you may be interested in assessing the number of likes, reviews, mentions, followers or fans. Diagnostic analytics can help you distil thousands of mentions into a single view so that you can make progress with your campaign.

#3: Prescriptive Analytics

While most data analytics provides general insights on the subject, prescriptive analytics gives you with a "laser-like" focus to answer precise questions. For instance, in the healthcare industry, you can use prescriptive analytics to manage the patient population by measuring the number of patients who are clinically obese.

Prescriptive analytics can allow you to add filters in obesity such as obesity with diabetes and cholesterol levels to find out areas where treatment should be focused.

#4: Exploratory Analytics

Exploratory Analytics is an analytical approach that primarily focuses on identifying general patterns in the raw data to determine the outliers and other features that might not have been anticipated using other analytical types. For you to use this approach, you have to understand where the outliers are occurring and how other environmental variables are related to making informed decisions.

For example, in biological monitoring of data, sites can be affected by several stressors. Therefore, stressor correlations are vital before you attempt to relate the stressor variables and biological response variables. The scatterplots and correlation coefficients can provide you with insightful information on the relationships between the variables.

However, when analyzing different variables, the primary methods of multivariate visualization are necessary to provide greater insights.

#5: Predictive Analytics

Predictive analytics is the use of data, machine learning techniques, and statistical algorithms to determine the likelihood of future results based on historical data. The primary goal of predictive analytics is to help you go beyond just what has happened and provide the best possible assessment of what is likely to occur in future.

Predictive models use recognizable results to create a model that can predict values for different type of data or even new data. Modeling of the results is significant because it provides predictions that represent the likelihood of the target variable—

such as revenue—based on the estimated significance from a set of input variables. Classification and regression models are the most popular models used in predictive analytics.

Predictive analytics can be used in banking systems to detect fraud cases, measure the levels of credit risks, and maximize the cross-sell and up-sell opportunities in an organization. This helps to retain valuable clients to your business.

#6: Mechanistic Analytics

As the name suggests, mechanistic analytics allow big data scientists to understand precise alterations in procedures or even variables which can result in changing of variables. The results of mechanistic analytics are determined by equations in engineering and physical sciences. Also, they allow data scientists to identify parameters if they know the equation.

#7: Causal Analytics

Causal analytics allow big data scientists to figure out what is likely to happen if one component of the variable is changed. When you use this approach, you should rely on some random variables to determine what's likely to happen next even though you can use non-random studies to infer from causations. This approach to analytics is appropriate if you're dealing with large volumes of data.

#8: Inferential Analytics

This approach to analytics takes different theories on the world into account to determine the certain aspects of the large population. When you use inferential analytics, you'll be required to take a smaller sample of information from the population and use that as a basis to infer parameters about the larger population.

Clean Up the Data

When used in reference to data, the term cleaning refers to the action of eliminating data points that are invalid from a given set of data so that the remaining data can be utilized as effectively as possible. Invalid points of data can either be those that are only partially available, are corrupted, or do not factor into the hypothesis that is being used to analyze the data at this point. It is difficult to remove the cleaning step from the realm of human judgment as the various variables that are being utilized to determine if a piece of data is relevant or not are often less black and white than those that a computer program could determine.

Furthermore, the points that are subject to cleaning are typically dramatic outliers of the data that you have collected. This means that they do not fit the flow of the other data that you have collected, often by being at one far end of the spectrum of data or the other. Determining which points are outliers is as easy as plotting the data and then looking for the points that are far away from the majority of the spread of data points. Alternatively, you can first run an analysis on the data in question before cutting out those points that are outside of the control limits that were set during the analysis. You can then remove those points and redo the analysis in order to get more accurate results.

The importance of having the cleanest, and therefore the most useful data possible cannot be overstated. It is common for analysts, especially those new to the field, to become somewhat lost in the complexity of the data that they are working with and the methods that are being used to analyze them. This, in turn, can easily lead to results that point in a misleading direction, causing hardship and potential financial ruin in the process. A good rule of thumb is that when you are analyzing statistics you are going to want to spend about 80 percent of your time ensuring that the data has been cleaned properly and the remaining 20 percent actually doing analysis.

Data Visualization

Sometimes just referred to as visualization, data visualization is all about creating visual representations of the data that you have discovered in an effort to not only explain the data in question, but also make it easier for anyone to understand. This is often done through things like statistical or informational graphics or by plotting data. Bars, lines and dots are all commonly used when it comes to visualizing numbers in an effort to make various hidden trends more visible. As it is not limited by the traditional confines of communicated information, a properly developed visualization will make it easier to find and explore new insights as well as understand the story that the data is trying to tell.

Making Effective Graphs

When it comes to graphing the information that you have found in an effective way, it is important to keep in mind the type of graphics that you will use in an effort to get to the heart of the data. In order to do that you are going to want to, first and foremost, show off the data in the most effective way possible by introducing those who look at the data to the substance of what it is you have been doing, not all the precise technical work that made it happen. Additionally, it is important to keep in mind that whatever you do, you are going to want to avoid accidentally distorting the data by presenting it in a misleading way.

Above all though, it is important that a good graph presents a large amount of numerical data in a small space that remains coherent, even when viewed from a large scale. It should also naturally lead viewers to compare various pieces of relevant data based on how it is arranged, and in so doing show the viewer data at numerous different levels of overall detail. You will also want it to be clear in its exploration, description, and purpose.

If you choose to go against these time-honored principles, then you are courting disaster in the form of misleading graphs which support actions that might very well be disastrous if followed through to their logical conclusion. Chartjunk is the phrase that is used to describe extraneous frills that novice analysts frequently add to their graphs in an effort to give them a little style or personality. This practice should be discouraged, however, as you don’t want anything taking the focus away from the numbers, their trends, and their relative importance.

Additionally, you are going to want to avoid adding anything that does not directly enhance the message the data is putting forth. This goes for separating related information so that the viewer has to keep turning their head back and forth in a gratuitous fashion as well; you want to cut down on everything that obscures the data in any way. Remember, when making a chart you are going to want to ensure that the amount of ink you use should be as minimal as possible while still transmitting the maximum amount of data possible.

Chapter 6 Commonly Graphed Information

There are eight primary types of information, sometimes called messages, that can be communicated via the data that you find, and each has a commonly used graph to make that message as clear as possible.

Time Series: In a time series, a single variable is going to be captured over a prolonged period of time. They are typically visualized with the use of a line chart in order to demonstrate their trend clearly. A good example of a time series is when the unemployment rate is tracked over a prolonged period of time.

Ranking: Typically done in either ascending or descending order, this information places categorical subdivisions in order based on an external, unbiased fashion. They are commonly represented by a bar chart which can be used to easily compare several different variables at once. A common use of a ranking system is when sales performance, which is being measured, is broken down by individual sales people, the categorical subdivision.

Part-to-whole: Typically, categorical subdivisions are measured in terms of ratio and the whole, generally in terms of percent out of 100. They are typically visualized via the use of a bar chart or a pie chart in an effort to making comparing various ratios as easy as possible. A common use for this type of visualization is the various market shares that certain competitors hold in a given marketplace.

Deviation: Categorical subdivisions can also be compared to a third party reference, as is often the case for things like budgeted for expenses versus the actual cost of expenses in a given timeframe. A bar chart is typically used in this scenario for the ease with which it can show comparisons between the two amounts in question.

Frequency Distribution: A frequency distribution is commonly used to indicate the number of times a particular was observed in a given period of time. The type of bar chart known as a histogram is typically considered the most useful type of graph to represent this type of analysis. Additionally, a boxplot can be useful when it comes to visualizing various statistics related to distribution including things like outlier, quartiles and median. Frequency distribution is commonly used for tasks such as determining the number of years a specific stock market return fell with a specific range.

Correlation: A correlation is useful when you need to compare observations related to two different variables for the purpose of deciding if they are likely to move in the same or the opposite direction. A scatter plot is generally considered the best choice when it comes to conveying this type of information. A common use for this type of visualization is determining how inflation and unemployment numbers track off of one another for a set period of months.

Nominal comparison: A nominal comparison is useful when you need to compare numerous categorical subdivisions but you aren't particularly picky when it comes to the order that they are in. A bar chart is the type of graph typically used when doing those types of comparisons and they are especially useful when graphing things like product code or sales volume.

Geospatial or Geographic: A geospatial or geographic visualization is useful when you are interested in comparing variables as they change over a physical space. This type of comparison is typically done via the use of a cartogram and can be used for things like determining the property tax in different areas of a state.

All told, when you are reviewing data, it is important to keep in mind the various ways that such data is commonly presented, and if one or more of the above visualizations is the right choice for the patterns that you feel are revealing themselves in the data that you are studying. If you find yourself in a scenario where there are multiple different variations of the data that seem to make sense when it comes to telling the type of story you are interested in telling, then the best thing to do is to simply generate both the visualizations in question and see which one expresses the message you are looking to convey in the simplest and most direct way possible.

Chapter 7 Data Visualization

Particular attributes of data could only be understood if they are presented in graphic form, see the Figure below. Data visualization or the art of representing data in visual form is crucial to transforming raw data into a sensible data that can be used by the business.

Figure: Data Visualization

To achieve real learning from data, business executives should know how to use visualization techniques to explore data and communicate the meaning of data to the rest of the business. In this Chapter, we will examine the process of visualization, as well as the people and technological skills needed for the business to make sense of data.

Businesses today are capable of capturing data at a fast rate mainly for reporting, compliance, and visualization. However, genuine value could only be achieved if the information has been processed, understood, and finally acted upon by the organization. Without an effective system for data processing and presentation, data is just data with no value for the business.

For many businesses, the development of data visualization technology followed a familiar journey: basic charts and tables done manually were replaced by Excel or Numbers, which was then succeeded by conventional business intelligence systems such as databases that can present information easily and as needed. These presentation features started as reports and were soon replaced by interactive platforms.

But with the rise of big data conventional business intelligence technologies may fall short if analysis, discovery, and visualization capacities are subtler. The market for data visualization has increased remarkably in the past few years as a way to provide insights into complex and large-scale datasets.

Basic data exploration tools are among the emerging areas of business intelligence, and the more conventional business intelligence software developers are innovating their products for the business user. Moreover, with the rise of powerful computers, smartphones, and tablets, people can interact with their data easily than in the past.

To put it simply, visualization capacities will enable the business to interact easily and understand big data. Data visualization is very effective in business because people are naturally attracted to visual analysis. We are highly-suited for identifying visual patterns and our brains are hard-wired for what we see to process better understanding.

Through data visualization, the business can also integrate large-scale of information in a single place, which allows people to make sense of numbers and text better.

Chapter 8 Four Important Features of Data Visualization Software

When you need to make millions of lines of data understandable in a short amount of time and to some end users, you need data visualization software to assist with this rather daunting task. Data visualization allows for understanding at a glance to allow technical and non-technical team members the same in-depth exploration and explanation of complex data patterns. Data visualization takes voluminous, coding-based data strings and turns them into easily understandable, interactive figures that speed implementation of necessary improvements to business processes. There are several things a business should look for in a data visualization program:

Point-Specific Reporting

The ability to provide a continual, synergistic method for analytics design that can show graphic representations of point-specific data allows a design method that immediately highlights issues within the data and allows technical and non-technical users the same level of understanding for the rapid implementation of solutions to increase operability and lower risk.

On-Demand Federation

If you want data visualization without constant dependency on an IT department, you need on-demand federation that takes into account multiple data sources and many formats. This allows businesses to avoid extra steps and time constraints necessary when using pre-analytic data consolidation or having to create new data warehouses to correctly position data for visual analysis. You want a versatile model that can access, read and use data across many dissimilar systems and formats to reduce overall cost and the risk of losing or misreading data while compiling it into graphical formats.

Incremental Scalability

Your business scope, needs and thus, data, changes rapidly and sometimes on a daily basis. The data visualization software you choose should provide a way to display data through all processes: from data discovery and on-the-fly analytics to automated controls. Software with the ability to scale incrementally allows for seamless transitioning among these stages within a single, user-friendly environment that can add processing capability only when it is needed for additional cost savings and attention to your bottom line.

Data Correlation

Most businesses choose data visualization software to help them manage business risk; discover and solve complex internal and external issues; optimize cash flow from current and potential business streams, increase positive customer experience and, ultimately, optimize operations across the board. To manage such broad tasks, a viable data visualization program will allow for data to be correlated across a process so that decision-makers can easily undertake a root cause analysis for any or all of these issues to provide rapid resolution and quick implementation of new processes or solutions and represented through the data. Cross-functional collaboration is encouraged through a graphical presentation that is understandable by all personnel, both technical and non-technical.

Areas of Data Visualization

To become a more effective tool, a business should invest in three important areas of data visualization: process, people, and technology.

Process

The system of developing data visualization such as a poster or an infographic involves a lot of discipline and may even require different sub-systems that should be well coordinated. The most essential step is to establish the ultimate goal of the data visualization right from the start.

People

Business owners should remember that people are closely related to the process. Hence, a wide range of skills is important to create an effective visualization. The skills needed for this are mainly drawn from data mining, statistics, computer science, graphics and also psychology.

Technology

With the range of technological options available in the market, selecting the right data visualization tool could be overwhelming, especially for business owners with no background in the area. This is the reason why you should first establish your goal to narrow down the wide categories of technologies available including business intelligence tools, graphic tools, analytics tools, and other tools.

In its raw form, data is just data, and it can only become valuable once you analyze it, make sense of it, and act on it according to its suggestions.

Hence, as the scale of data being captured increases, so does the need for visualization to be understood and communicated effectively.

Businesses should invest in the capacities to learn from the data through visualization. But before you can do this, you should understand the whole importance of the process, people as well as the technological requirements. These three areas are integrated with one another, so the business should work effectively to combine these areas to work on one goal.

Experts believe that in the future, data visualization will become more interactive, real-time and accessible by everyone in the business. As new technologies emerge and data visualization becomes more advanced, business executives should be able to respond faster and better.

And as data becomes more interactive, it will be easier for anyone to explore the numbers easily without exerting too much effort or background in data science. The business can improve interactive and real-time data visualization so it can leverage on the platform to reach more people such as customers, partners, stockholders, and so on.

Furthermore, improvements in technology could make visualization a lot easier to develop, which could allow more people to access visualization. Your business should continually develop its systems to adopt visualization as a mode of communication and data exploration.

This will not only enhance data visualization improvement in the long-term. This will also offer businesses with the information and understanding they should learn from their data and act accordingly.

Advantages of Using Data Visualization for Business

One important attribute of a competitive business is the ability to make decisions as fast as possible. But before a business can achieve that, business leaders should have access to data in real-time and interpret it accordingly.

Gathering information and the capacity to understand it is crucial for businesses. To detect business opportunities earlier that competitors, the business must be able to access, assess, understand, and respond to information faster and more effectively than ever before.

The tools and strategies for data visualization can enable the business and its team to work on new strategies to significantly enhance their skills to grasp data hiding in the layers of information. Below are the top five advantages that data visualization can provide to business owners.

Visualize patterns and connections between operations and sales and marketing activities

Among the important advantages of data, visualization is how it can enable business users to more easily detect relationships as they happen between sales and marketing performance and business operations. In the highly competitive nature of the business world, looking for these relationships within the data has never been crucial.

For instance, let’s say a business executive for a toy company is reading the monthly customer information. The executive can access a bar chart, which shows that the net promoter score of the business has decreased by three points in the past quarter in Japan. The report suggests that there is a concern with customer satisfaction in the country. However, the report lacks on insight to explain why the ratings got dipped.

By offering a more comprehensive report of the business as well as operational dynamics, data visualization allows the leaders to see that the recent events in Japan have affected the customer satisfaction. The capacity to make such connections will allow the business leader to determine the real cause of the concern and respond faster for resolution.

Make Sense of Data in New and More Innovative Ways

According to a research conducted by the University of Pennsylvania School of Medicine, our eyes can transfer information at about 10 mbps, which is about the same speed as internet connection. However, many business reports are presented for business executives who may fail to understand the data because these are filled with plain charts and static tables that cannot supplement the information.

On the other hand, data visualization allows end users to absorb a large-scale information about business and operations. Effective visualization will allow business executives to identify relationships between multi-faceted data sets and offers better ways to make sense of data by using fever charts, heat maps, and other helpful and relevant graphical representations.

Businesses that are using visual data are more capable of finding the information they need once they need it and can do so more effectively compared to their competitors who are not using data visualization.

According to a market survey conducted by Aberdeen Group in 2013, business managers who are using visual data tools have 28% more chance to find relevant information compared to their peers who are relying on dashboards and traditional reports. Furthermore, about 48% of business intelligence users are businesses who are using visual data can find information they need even without asking the help of their IT personnel.

Detect and Respond on Rising Trends Faster

The scale of information that businesses are capable of capturing about their customers as well as market trends can offer business executives with relevant insights into new opportunities to do business and increase revenue as long as they can spot on opportunities in the layers of data. By using effective data visualization, business executives can detect sudden changes in customer behaviors as well as market conditions across several data sets much faster.

For instance, marketing executives for a grocery chain could use data visualization to see that customers are willing to spend more as the economy improves and they are also more interested in buying organic items that are more expensive compared to regular items.

A more comprehensive analysis of customer behavior and other data sets suggest a rising opportunity for the store to launch a special section of organic products. Business insights like this can enable the organization to respond to this new business opportunity ahead of its competitors.

Tell a Story

Another benefit of using data visualization for business is its capacity to tell a story that everyone in the business and its stakeholders can relate to.

For example, business executives for a manufacturing company who are monitoring crucial indicators like net profit margin and EBITDA. They could only view a part of the story pertaining to the current conditions of the business through a bar chart, which may not show that the business already produced 2% revenue growth for the previous month. This report doesn’t show which specific categories are increasing or decreasing as well as the probable reasons why.

A heat map can show which product categories are performing or underperforming, and will enable business executives to consider the data to figure out the factors that are shaping sales. The data could reveal that cosmetics are not performing, but that higher-income segment comprises the majority of sales. The business can use this insight to target marketing efforts to this customer segment to increase conversion rates as well as profit growth in this category.

Communicating business executives with data visualization could open better ways of looking at operational and business data, which enables senior management to find new opportunities while enabling a wider audience of analytics with the goal of advancing the business.

Directly Interact and Control Data

Among the top advantages of data, visualization is how it could provide actionable business insights. Unlike static charts and tables, which you can only view, data visualization will allow users to control and interact data.

For instance, a traditional business report can inform a business owner of an appliance company that sales for coffee makers for July are down. But the report cannot inform the owner why sales of coffee makers are down or if specific brands or price are doing better compared to others. Furthermore, the data contained in the report might represent the situation days or weeks ago, so the data is not accurate and does not signify the current trend.

Through real-time predictive analytics integrated with data visualization, the business owner can view current sales figures and check why specific brands are not performing well as well as the possible reasons that revenues are lagging behind - for instance, sales campaigns launched by competitors.

The business owner can identify the best action according to the analytical models that can be developed specifically for the business. For instance, he may launch a month-long sales promo for certain dealers that are targeted at the most possible buyers with a price that can undercut the competition but can generate acceptable profit.

Chapter 9 Big Data Impact Envisaged by 2020

Have you realized that it is not just advertising and marketing that organizations are taking online these days? If you consciously think about what takes you online on a daily basis, you will realize that a good part of it is business, often because you feel you can find a wider choice of items there, plenty of information about the product or service that you want, and even a wide range of pricing. The web also seems to provide much more entertainment than you could physically reach in a short time and, most probably, at a relatively smaller price. When it comes to connecting with others, individuals, private and public institutions, and all manner of organizations are taking their communication online, where they can reach a wider audience much faster and more cheaply.

How Does Moving Online Impact Data?

First, it means that the amount of data being generated on a daily basis is growing exponentially, and we can no longer ignore big data. Second, even before anyone can speak of the ability to analyze and organize the massive data, tracking is already a mammoth challenge in itself. Would you believe that internet users are generating data in excess of 2½ quintillion bytes each day? This includes automated feedback, such as traffic monitors, weather-related trackers, and all manner of transactions. Is this a good thing?

Well, potentially, it is. However, there is the question of what sources are reliable and which ones are not, what data is relevant to your scenario and which one is not, and so on. As you may already know, having massive data before you can also be overwhelming. That is why big data has created room for a unique business, where you get people with special training to make good use of big data. That potential power of big data is what specialized companies seek to help you unravel and take advantage of by handling big data that affects your organization.

In fact, specialized technology has started to make big data management convenient. A good example of this is Apache™ Hadoop®, an advanced database management technology that takes you beyond the consolidation of information towards the improved efficiency of your business and increased profits. As long as organizations are open to the use of advanced technology that makes big data analysis and management convenient, the world is headed for better times.

Instead of being overwhelmed by high data traffic and an accumulated mass of data, organizations are going to unleash the power of such data, and if they are in education, they are going to drastically bring down the cost of education. If they are in meteorology, they are going to have better comprehension of certain complex phenomena, such as the weather. Those in business will be able to raise their productivity by drastically cutting on all manner of redundancies, and the job market is going to have better correlated data sets that will help to match job seekers with potential employers based on the skills offered and needed, respectively. Some of these things are already happening, and they can only become better.

The story of big data does not end with the reduced cost of doing business and acquiring education, increased efficiency, and productivity, as well as increased profits. Ongoing research points to the potential of improving the fight against crime, significant improvement in web security, and the ability to foretell the likelihood of an economic or natural disaster sometime in the future. In fact, as long as there is ongoing work regarding big data and its intricacies, the world is likely to witness bigger and more radical changes, much of it for the better.

What's the Market Like for Big Data?

As previously explained, there are many innovators trying to create analytical and data management tools that will turn massive data into an economic advantage. In 2012, the big data market was worth $5 billion. If you remember what we said earlier that big data has been growing exponentially, you will not be surprised to know that if the trend today continues, the market for big data is projected to reach $50 billion by 2017.

Specific Ways Big Data Can Significantly Impact Life – Soon

1. Websites and applications functioning better

In this regard, it is expected that it will be much easier and more convenient to navigate websites and make sense of data, but and these sites are believed to become much safer than they are today.

This is particularly so because big data can be useful in identifying and tracking fraudulent activity on a real-time basis. Big data can introduce fresh and clear visibility on an organization's website while making it possible to foretell when attacks are imminent. Innovators have actually already begun designing programs geared towards safeguarding data against destructive intrusion. For example, there is the machine learning program known as MLSec that you can find at MLSec.org. It uses algorithms under supervision to locate networks harboring malicious programs. It must be very encouraging to web users to learn that this machine learning program has been proven to be accurate at a rate of 92% to 95% for every case tested.

How Bad is the Security Situation Today?

Based on 2012 statistics:

- Of all the websites hacked in this year, 63% of the owners did not realize they had been invaded.

- In fact, 90% of web owners did not even seem to notice anything strange at all going on within the site.
- Notably, 50% of the web owners learned that their websites had been hacked from their browser warnings or even warnings from the search engine they were using.

2. Higher education becoming more accessible

Why is it so exciting that cost of education would fall? Well, accessing higher education is generally a problem for the average person in most countries. In the U.S., which is considered by many as the land of plenty, the cost of tuition rises at a rate that is double that of healthcare. When you compare the hike in the cost of education to that of the country's Consumer Price Index, it reaches four times as high.

Luckily, just as websites are becoming better designed and protected, something is happening to make education more easily accessible to a larger number of people. A good example is the emergence of online sites offering great courses. The Khan Academy found at khanacademy.org, Big Data University found at BigDataUniversity.com, Venture Labs found at venture-labs.org, and Coursera found at courser.org are just some examples of institutions that are offering higher education online at a much lower cost – and sometimes even free of charge – than conventional tertiary institutions.

The good thing about many of the courses offered in this manner is that students are tested on how well they have clinched the skills taught, particularly because these skills are applicable to the current high-tech environment. For example, Big Data University teaches Hadoop, along with some other big data-related technologies.

3. Relative ease in getting a job

Have you ever given thought to the number of job seekers there are worldwide? It must be a staggering figure. In fact, on a monthly basis, the number of job searches on the web alone has hit 1.5 billion. On the positive side, there are websites that have already begun to match job seekers with potential employers by amassing valuable information on various employers, as well as information on interested job seekers. One such website is indeed.com.

4. Improved road safety

Are you aware that car accidents are the main cause of death in the category of youth between 16 and 19 years of age in America? While it is dangerous to drive while under the influence of either drugs or alcohol, 75% of the deaths in this category are not related to alcohol or drugs. What this means is that there are many accidents that occur merely because of poor judgment. It is envisaged that as scientists continue to work on advancing the technology being used on computers, big data is going to help them predict the behavior of drivers on the road, as well as the move a vehicle is about to make at a certain point.

This route is geared towards reaching a point where cars on the road can exchange data so that drivers in different cars can see up to three cars that are ahead of them, three that are immediately following them, and three on either side of them at any one time. In fact, it is said that big data will enable the drivers in data swapping vehicles to see the posture and focus of a driver in a car near theirs. This may sound a little far-fetched, but think about it: Haven't Google's self-drive cars taken the auto industry to a totally new level?

5. Ability to predict business future

What this means is that organizations will be able to use software like Hadoop to analyze the data at their disposal in a way that will bring future possibilities to the fore. The speed and accuracy with which data will processed will enable organizations to take prompt action where there are business opportunities emerging, where damage control is needed, and where other actions that call for accurate assessment are required. There are already big organizations using Hadoop with great success. These organizations include eBay, Twitter, Facebook, Disney, and others. The demand for Hadoop is rising rapidly. IDC, a renowned market research firm, has predicted that by 2016, the conservative worth of this software will be $813 million.

Another good example is Recorded Future, a technology company based on the web. This organization provides data analysts with security intelligence that they use to keep their information safe. It puts businesses in a situation where they can anticipate risks and also capitalize on business opportunities by unlocking predictive signals using clever algorithms. There are other examples already helping businesses prepare for eventualities, but suffice it to say that as technological development continues, it will become all the more possible to leverage data, hence avoiding surprises.

6. Ability to predict matters of weather

The ability to predict the weather brings the advantage of being in a position to protect the environment better. Such protection, in turn, brings in huge savings to the country and the world at large, considering the massive expenses that are brought about by weather-related disasters. For example, weather- and climate-related disasters cost the U.S. losses in staggering figures that are actually in excess of $1 billion. Here, we are talking about such disasters as drought, wild fires, incidences of flooding and storms, and other unfortunate events.

Things are already looking bright on the data technology front. For instance, there is the Joint Polar Satellite System (JPSS), which is set to be launched in 2018. The JPSS, which will have the capability to utilize sensor technology, will also use data to determine a hurricane’s path or a storm’s path well before these disastrous events occur. This will then give everyone concerned time to plan what to do to safeguard life and property. As CNBC News has already noted, situations that relied on guesswork some years back are slowly but surely becoming something to predict with precision through predictive science.

7. Healthcare becoming all the more efficient

Big data is expected to bring improvement to the health sector, not just by raising efficiency levels in service delivery but also by customizing services to suit respective consumers. McKinsey & Company, an advisor to the management of many renowned businesses, says that between 50% and 70% of business innovations depend to a great extent on the capacity to capture customer's data and not as much on external analytics. McKinsey & Company relies heavily on qualitative and quantitative analysis to be able to give helpful advice to management.

It is said that 80% of data in the medical sector is unstructured. However, with the trend the health sector in the U.S. has taken in using big data creatively, great improvement in service delivery is anticipated. It is actually envisaged that big data is going to help the sector increase value through efficiency, adding value in excess of $300 billion each year. By the same token, expenditure is anticipated to be reduced by a good 8%.

In treating patients effectively, caregivers benefit a lot from available patient data. This is because such data helps caregivers provide evidence-based advice. Currently, a medical center within Boston called Beth Israel Deaconess is putting a smartphone app in the market as a means to help medical caregivers access 200 million data points. These data points are expected to make data concerning around 2 million patients available. There is also Rise Health, which utilizes the accessible mass of patient data by analyzing it from all dimensions and aligning it with the health providers' goals to improve healthcare through fresh insights. Overall, big data brings speed to innovation. A fitting example is the project on the human genome that took 13 years to complete; today, it would only take a couple of hours to accomplish it.

Chapter 10 Pros and Cons of Big Data Analytics

As there is so much data so, the organizations need to collect and store them. The data becomes valuable to businesses when it is analyzed. The word big data not only describes a big data sets but it also relates to the techniques, frameworks, and tools to analyze the data. The data can be gathered with the help of social media, search engines, infrastructure and public utility. Big data has both advantages and disadvantages which are as follows:

Pros:

1. There is a possibility to store large data.

2. Big data is usually stored in clouds, so they are readily available from any place.

3. Big data has a high speed during processing and transmission.

4. In the past, data volume was limited, and processing tools were weak. Today, technologies, tools and analytical methods are modern which allowed analysts to gain a lot from the big data.

5. It detects errors and fraud quickly. The criminals are caught red-handed with its safe-guard system when attempts are made to hack in an organization. The department of IT security takes immediate action.

6. The big data tools can be expensive, but it will finally save a lot of money. The company’s overall burden is also reduced.

7. The data that is collected is highly valuable. The big data analysts can tell an organization that which product is performing right in the market and which is not, so that an organization manufactures that product more which is appreciated a lot by customers, leading to real revenue.

8. Real-time big data get into the errors quickly which helps companies and organizations to limit the effects of a problem, and they can stop the decline of customers from purchasing their products. So, the problems inside the organization are caught instantly.

9. While competition with other companies, you stay ahead in our competition as you will get a notification when your competitor will change the strategy or lower the prices.

10. With the help of big data, the services are improved. Organization became very active in responding to the failures of their products. e.g. they have attached a sensor in their vehicles when the car requires maintenance, the sensor gives the notification.

11. The company becomes trendy according to the needs and demands of the customers. The company keeps in touch with the offerings of its competitors, and the movements of the customers give a valuable information about the latest consumer trends. Quicker decisions are made with the help of big data analysts.

Cons:

1. Several meaningless data points might be present in the big data, so analyst needs to be careful in separating the meaningless points from the essential points.

2. Big data also has privacy problems e.g. the data of a student's academic performance can easily be seen by every person on the internet.

3. The security level of big data is very low.

4. Companies that hope to utilize big data will require amending their whole method as the data that is rolling into the company is continuous rather than episodic.

5. Most of the data tools that are used currently are unable to handle analysis of real-time.

6. The real-time big data analysis needs new tools to be used. Presently for the analysis, the Hadoop standard version is not suitable, so it requires computer's special power.

7. It requires a different style of working in your organization when you are using a real-time big data. It is common when an organization typically receives insights once in a week, but when an organization receives insight after every second then the organization will have a different approach and the method of working will also be different.

Tools Used by Big Data Analytics:

Presently, Hadoop is not offering a real-time big data analytics; other tools need to be utilized. Let see the following tools that perform a job very well.

a. Storm:

It is a real-time big data system which is owned by Twitter. It's working method is the same like Hadoop. It is very easy to use the storm as it tolerates the faults/errors.

b. Cloudera:

The tools of the Cloudera company offer a real-time big data analytics. Data is stored in HBase or HDFS which serves as an integral part of Cloudera Impala.

c. Gridgrain:

The gridgrain is a company for grid computing made for Java. Big data analytics use gridgrain as a tool for real-time data.

d. SpaceCurve:

SpaceCurve is developing a technology which has the ability to discover original designs in geodata, which is different as compared to the normal data of mobiles. Their big data platform and tool set has created a new world record by running the demands at a speed of 10GB/sec.

The Big Data Future:

Big data has become a priority for the businesses and government agencies. Every company needs to evaluate whether the pros of the big data outweigh the cons or not. The strategy would be different for every business.

Big data demands close attention to the changes. Individual should have open eyes and strong intuition while carrying a big data for his business. There are a lot of companies, corporations, organizations which want to store, analyze, and visualize a data in real-times but just to pour a lot of data in an organization is totally a different thing.

Conclusion

Prior to the recent rise in analytics, businesses and organizations did not have the capacity to analyze a great deal of data, so a relatively small amount was maintained. In today's data-driven world, anything and everything may have significance, so there has been an attempt to record and keep virtually any data that we have the capacity to collect; and we have a great deal of capacity. Beyond the quantity of data that we are gathering and storing is the quality of the data. That is to say, data has grown beyond basic facts and figures to encompass media files. Video, audio, and presentations have all become units of data for possible analysis. A major concern with regards to data analytics is how to store and maintain all of these rapidly-increasing piles of data. The data science community has begun to rely more heavily upon the software engineering community, in order to find solutions to our over-abundance of data.

Not all data is necessarily valuable. Society now has advanced data analytics that allows us to glean useful and important information from even the smallest bits of data. Such information, when reconciled with other groups of information, can (and has often) resulted in breakthrough of modern science, business, and economy. As we consider our need to increase the role of data analytics in the ways that we function as organizations, we should keep in mind that data does not contain all of the answers to our growth and advancement. Data provides us with the building material with which we can create new understanding and innovation. The other part of the process is distinctively human. This part includes creativity, risk taking, and cooperation. It appears as though the less we have of one, the more we need of the other. The more intellectual rigor and collaboration between various fields of science the more that we seem to benefit for even limited amounts of data. Conversely, the less of those things that we have, the more data we need in order to learn, grow, and innovate. Perhaps, the solution to our looming problem with big data is to reduce our need for so much of it.

Deep Learning with Keras

Introduction to Deep Learning with Keras

By Anthony S. Williams

Table of Contents

Real World Applications

Conclusion

Introduction

The primary goal of this book is to introduce you and show you the significance of one of the most powerful and the most popular libraries for creating deep neural networks in Python. In the next sections of the book various deep-learning algorithms in Keras will be explained, as well as step by step guides on how to build convolutional neural networks in Python.

It should be noted that this book is geared toward beginners who are just beginning to explore the power of such a library and its significant influence on network development.

So, after you are done with reading, you will be able to create your first neural network model using Keras, you will get familiar with the practical usage of Keras, and its implementation and you will see what are some of the real-world examples of Keras and what is its overall influence in deep learning studies.

Keras is recommended library for deep learning in Python since it is minimalistic, it is capable of running on top of Theano, Deep learning, Tensorflow, and CNTK. Keras is carefully designed to make fast experimentation possible with deep neural networks. The focus of the Keras design is to keep it modular and extensible.Developed as a part of ONEIROS, Keras's primary author is François Chollet who is a Google engineer. It was introduced in 2015, and its author later explained that Keras was developed to be an interface rather than machine learning framework. Keras, in fact, presents a higher-level intuitive set of abstractions which make it easy to configure neural networks.

Keras Features

The main features of Keras are implemented from commonly used neural network building blocks like objectives, layers, activation functions and optimizers. With Keras it is easy to work with text and image data since it features a host of tools suitable for any beginner.

Facts are speaking for itself, Keras is the second-fastest growing framework, and that’s not surprising. Keras is used to solve problems without having to interact with the TensorFlow or Theano, which are underlying backend engines. So, without any interaction with the background, end-to-end problems can be solved.

Initially, Kears was developed on top of Theano, but shortly after the release of TensorFlow, Keras added it as backend. As new generation computation graph engines are designed, Keras will extend to support those as well in the future. Keras is not dependable on backend engines since it features its built-in graph data structure which can handle computational graphs. Therefore, it doesn’t rely on TensorFlow’s or Theano’s native graph data structure.

As a result of this independent data structure, Keras can make offline shape inference in Theano which is missing, but very much needed feature in Theano. Easy model copying and sharing are also doable in Keras.

Since it is independent of backend engines, Keras manages all its graphs independently, so it is possible to define any Keras model with the Theano engine backend. It is even possible to switch to the TensorFlow backend or to re-apply your Keras model which is Theano built on a TensorFlow input which creates a TF model version. This TF model is what was a Theano model initially created.

Another important feature of Keras is an object-oriented design. Therefore, in Keras everything acts like an object including models, optimizers, and layers. All parameters regarding any object can be accessed. A different approach would be as in other deep learning models which are defined as chains of functions since most of these functions are heavily parameterized by the weight of their tensors. Since the manipulating of these parameters would be impractical in a functional way, Keras approach is to treat everything like an object.

If the situation is different, the functional approach will imply layers, models, and optimizers as functions which would create more weights when called, and they would be stored in global name-indexed collections with other models which are using this approach. For example, TensorFlow has taken this approach. This method is just impractical since many operations like accessing an existing tensor or simple model loading must be approached by name-matching.

Therefore, every tensor you create you must name carefully instead of relying on automatically generated names. With this method, there is always a risk of name collision which can prevent you from being able to manipulate multiple models in a single session.

Chapter 1 Deep Neural Network

Deep learning, which is also called as hierarchical or deep structured learning, is the application for learning chores of neural networks which are artificial denoted as ANNs.

Artificial neural networks contain multiple hidden layers. Being a part of the greater family of machine learning techniques, deep learning is based on learning data representations in opposition to task particular algorithms. Deep learning can be unsupervised, partially supervised and supervised.

Deep learning models like recurrent networks and deep neural networks are applied to broad range of fields including, natural language processing, bioinformatics, machine translation, computer vision, etc. Deep learning models are classified as learning algorithms machines.

The methods of deep learning include usage of multiple layers of nonlinear processing components in order to extract and transform their features. Every successive layer is based on the output of the previous layers which is used as input.

Other deep learning models which are unsupervised are based on learning of numerous levels of representations and features of some image or text data. To form a hierarchical formation, features from higher levels are developed from the lower level features. Also, deep learning models can present the correspondence between numerous levels of features and different abstraction levels which are forming hierarchical concept.

The common features you can find between these different deep learning models are multiple layers that are regarding nonlinear processing components and both an unsupervised and supervised learning of representations and features of each layer. The concept is always hierarchical from lower to higher level features. Depending on the problem which will be solved, the composition of multiple layers vary. These multiple layers include collections of propositional formulas and hidden layers of a neural network.

Deep learning was introduced and designed by the World School Council London that has been using certain algorithms to alter their inputs over multiple layers rather than from shallow learning algorithms. The signal is transformed from each layer by a processing component just like an artificial neuron.

There are different paths of signals from every layer. Credit assignment path regards to any chain of transformations from each input to output, and causal connections which potentially may occur are described as well.

Another way is through Cap depth when the total number of hidden layers is always plus one since the output layer is also given in certain parameters. For recurrent networks, the Cap depth can be unlimited, since each signal may transfer through each layer multiple times. Deep and shallow learning refers to a threshold of shallow depth learning developed from deep learning.

Distributed representations of layers are observed as all data which is generated by the interaction between layered factors. Deep learning is based on the assumption that each layer of factors is in a correspondence to the other levels of composition and abstraction.

The number of created levels of abstraction depends on layers size and number. The hierarchical concept of deep learning refers to that abstract concept and higher level components which are derived from the lower level ones.

Deep neural networks may include few fundamental approaches. Each deep learning model can be successful in certain domains, but the comparison between models isn't always possible unless the models have been estimated on the same data collection. Deep neural networks are feedforward meaning that data is transferred from an input to output once, without looping back.

On the other hand, the reccurrent neural network makes it possible for data to move in any direction which is commonly applied to language modeling. About certain interpretations and applications of deep neural networks will be the word in the later sections of the book.

Neural Network Elements

Neural networks are a collection of algorithms created to recognize certain patterns. We are already familiar with these networks and how they transform data through created model of perception, clustering, and labeling raw input. The patterns which are recognized are numerically denoted as vectors, and all real-world data including text, images, the sound must be translated into those vectors.

Keep in mind that neural networks are classifying and clustering different data. Think of them as a certain layer which you put on top of your data that you managed and stored.

The neural network works for you as a help in order to group and classify different data in a correspondence to similarities between example inputs. Think of deep neural networks as a part of the greater machine-learning application for regression, reinforcement learning, and classification.

Deep learning is used for networks which are composed of multiple layers. Each layer is made of a node which is a place where computations occur. These nodes from every layer combine input from a data with a collection of coefficients, or certain weights, so the significance is assigned to every input for each task the algorithm is processing or learning.

Therefore, these input components which carry certain weights and features are later summed, and that certain sum later passes through particular node which is called as an activation function. Further an extent of the signal is determined, which affects an outcome of classification.

Each node is like a neuron switch which can turn off or on as an input is transferred through some net. Starting from an initial layer, each layer's output is derived from it and simultaneously transfers between nodes. To assign significance to every layer's features, individual weights must be paired to initial input layer. This is how the network clusters and classifies any input.

Types of Deep Neural Networks

There are various types of deep neural networks including Hopfield network, deep belief network, generative adversarial network, autoencoder, radial basis network,etc.

However, in this book, we are going to see how to create different types of neural networks with Keras such as recurrent neural network, deep feed-forward network, deep convolutional networks and de-convolutional network .

But firstly, some details regarding these types of deep networks should be discussed, before going further into the subject.

Feed-forward neural network is where connections among different components do not form a cycle model, which would be the main difference between feed-forward and recurrent neural networks. The feed-forward network was, in fact, the initial and the simplest neural network model.

The information contained in this network are being transferred only in one linear direction, forward, from the initial input nodes to the hidden layers and their nodes. As mentioned before, recurrent neural networks form connection among components in the form of cycle. This model enables a network to expose dynamic temporal behavior. As for a convolutional neural network, it is a type of feed-forward network which has already proven value while analyzing visual imagery.

Chapter 2 Keras Models

Keras is a high-level, among the most popular neural networks and a great alternative to TensorFlow and Theano. Being written in Python, it is capable of processing and running on top of CNTK, Theano, and TensorFlow. Keras deep library is enjoying great popularity since it is created for fast and easy prototyping, it runs smoothly both on GPU and CPU and supports both recurrent and convolutional networks, as well as combinations of these two.

The basic structure of Keras is a layer model which is a way of organizing multiple layers. The simplest model of these layers is the sequential model, in other words, a linear processing of layers.

Keras Sequential Model

From keras models.import Sequential

model = Sequential

To execute more complex models, Keras functional API is used, which is capable of building entirely arbitrary graph of layers.

The simplest sequential model would look like this:

from any keras import model

model is equal to sequential ()

To create more complex Keras models, stacking layers are added with .add () which would look like the following:

from any keras model to layer import and activation

the addition of a model with dense value

model activation

Stacking Layers in Keras

```
from keras . layers import Dense, Activation
```

```
model . add ( Dense ( units= 64 , input_dim= 100 ))
model . add ( Activation ( ‘relu’ ))
model . add ( Dense ( units= 10 ))
model . add ( Activation ( ‘softmax’ ))
```

When you complete your model, you have to configure its process with .compile as shown below:

```
model.compile ( loss = ‘categorical_crossentropy’ ,
          optimizer = ‘sgd’ ,
          metrics = [ ‘accuracy’ ])
```

A fundamental principle of Keras is to make things as simple as possible and let a user be completely in control when needed. If you need to, your optimizer can be configured further as shown here:

```
model.compile ( loss = keras . losses . categorical _ crossentropy,
          optimizer = keras . optimizers . SGD ( lr = 0.01 ,
momentum = 0.9 , nesterov = True ))
```

compiled model

the optimizer is equal to keras

If needed you can also manually feed batches to created model or evaluate performance in just one line where loss and metrics would equal to model evaluation.

Feed Batches in Keras

```
Model . train _ on _batch ( x _batch, y _ batch)
```

Sequential Model

Predictions on new data also can be generated where classes would be equal to model prediction. When it comes to the sequential models in Keras, the model must be familiar with input shapes and what to expect. Therefore, the initial layer in a model has to receive data information about input shapes. This has to be done only for the initial layer since the following layers will automatically share interface based on the initial layer.

Therefore, an input shape will be passed to the initial layer which is known as a shape tuple or tuple which contains integers or no entries at all. None entry is the indication that only positive integers could be expected. The batch size and dimensions are not included in a input shape. Other layers like 2D layers can use dense which support a certain specifications of their initial input shape.

Creating a Sequential Model in Keras

```
from keras . models import Sequential
from keras . layers import Dense, Activation

model  = Sequential ([
   Dense ( 32 , input _ shape = ( 784 ,)),
   Activation ( 'relu' ),
   Dense ( 10 ),
   Activation ( 'softmax' ),
                          ])
```

the argument for 2D layers which support input shape specifications is

input _dim

and for 3D layers same support is

input _dim and input _lenght

If needed and you have to specify a batch size which is fixed for your inputs when used for recurrent networks, you just have to pass an argument batch size to a certain layer. Before creating any training model, the configuration of the learning process is mandatory and compile method is used.

```
model = Sequential ()
model . add ( Dense ( 32 , input _ shape = ( 784 ,)))
model = Sequential()
model . add(Dense( 32 , input _ dim= 784 ))
```

Three arguments will be received as loss, function, an optimizer and a list of metrics. An optimizer is a string of identifier of an already existing optimizer. The second argument is a loss function which is also a string identifier of an already existing loss function. The third one is a list of metrics, a string of identifier of already existing custom metrics functions.

optimizer arguments: Adagrad or Rmsprop

loss function arguments: use or categorical cross-entropy

list of metrics argument: metrics equal to the accuracy

Classification Problems

For a multi-class classification problem

```
model.compile(optimizer= ‘rmsprop’ ,
          loss= ‘categorical_crossentropy’ ,
          metrics=[ ‘accuracy’ ])
```

For a binary classification problem

```
model.compile(optimizer= ‘rmsprop’ ,
          loss= ‘binary_crossentropy’ ,
          metrics=[ ‘accuracy’ ])
```

For a mean squared error regression problem

```
model.compile(optimizer= ‘rmsprop’ ,
          loss= ‘mse’ )
```

For custom metrics

```
import keras.backend as K

def mean_pred(y_true, y_pred):
    return K.mean(y_pred)

model.compile(optimizer= ‘rmsprop’ ,
          loss= ‘binary_crossentropy’ ,
          metrics=[ ‘accuracy’ , mean_pred])
```

Many problems can be solved by this method including multiple classes classification problems, binary classification problems, error regression problems, custom metrics, etc. When it comes to the training models in Keras, they are created on Numpy arrays of labels and data od initial layer. Mostly you will use .fit function for training models.

For a single-input model with 2 classes (binary classification):

```
model = Sequential()
model.add(Dense( 32 , activation= ‘relu’ , input_dim= 100 ))
model.add(Dense( 1 , activation= ‘sigmoid’ ))
model.compile(optimizer= ‘rmsprop’ ,
              loss= ‘binary_crossentropy’ ,
              metrics=[ ‘accuracy’ ])
```

Generate dummy data

```
import numpy as np
data = np.random.random(( 1000 , 100 ))
labels = np.random.randint( 2 , size=( 1000 , 1 ))
```

Train the model, iterating on the data in batches of 32 samples

```
model.fit(data, labels, epochs= 10 , batch_size= 32 )
```

Therefore, you can create a model which is consisted of two classes, binary model, and you can generate dummy data or train the model which is consisted of numerous batch samples. Categorical classification is also possible and single-input model consisted of 10 classes can be trained.

Functional API Model

To define more complex models like models which generate multiple outcomes, models with shared layers and acyclic graphs, Keras functional API is the way to achieve this. So, as you are already familiar with the simple sequential model, creating more complex model won't be a problem.

A densely-connected network is better to create a sequential model, but to learn basics of more complex models, it's better to get familiar with this. Each layer instance is on a tensor or callable, which means it returns to its tensor. Both output and input tensors are used to define a complex model.

Just like sequential models, these models also can be trained in the same manner. To return to tensor inputs are equal to input. Like we mentioned, each layer distance is callable meaning it returns to a tensor, so predictions are equal to dense. If you want to create a model which would consist of three dense and one input layer, as already mentioned .fit argumentation must be used.

Starts model training

Model . fit (data, labels)

return to a tensor

inputs equal to input shape

a layer instance return to a tensor

predictions equal to dense

the model consisted od three dense and one input layer

model.fit including labels and data

In fact, all models are callable, meaning you can treat any model as if it is a layer, by returning it to its tensor. Wit the functional API, it is straightforward to use already created models. However, it should be noted that by re-using models, you are not just calling a model's structure, but as well as its weights. This makes it easy to create models quickly, and these models can process previous sequences of inputs. Re-using of previously created model structure and its corresponding weights would look like this:

```
x = Input ( shape = ( 784 ,))
y = model (x)
```

x model = input

y model = model x

For example, using this method, you can transform an image model to video classification model using only one line.

```
from keras . layers import TimeDistributed

input _ sequences =  Input(shape=( 20 , 784 ))

processed _ sequences =  TimeDistributed (model) (input _
sequences)
```

Multi-output and Multi-input Models

Models with multiple outputs and inputs are those used for any functional API. Manipulating with a significant number of datastreams which are connected is easy with the functional API. For a better understanding of models with multiple outputs and inputs, an example will serve the best. Therefore, let's assume we want to predict how many likes a news headline can receive. Therefore, a primary input of this model will be the headline, in this case, a news headline posted on the Internet.

Following behind the headline, there will be words in a sequence, but to make things a little bit complicated, this model will include an auxiliary input, in other words, it will be able to receive extra information like when the headline was posted, etc.

Also, the model will be supervised by two loss functions, and the main loss function will be based on previous one which is already created. Good regularization of any deep model is to use the main loss function earlier while creating a model.

So, by using the functional API, this model can be implemented. Therefore, the main input will receive the data in this case news headline, which will be a certain sequence of integers.

Each of these integers encodes a particular word, and the value will be between one to ten thousand, in other words, integers are a vocabulary containing ten thousand words. Sequences in the model will be a hundred words in length. It will look like this:

```
from keras.layers import Input, Embedding, LSTM, Dense
from keras.models import Model

main_input = Input(shape=( 100 ,), dtype= ‘int32’ , name=
‘main_input’ )

x = Embedding(output_dim= 512 , input_dim= 10000 ,
input_length= 100 )(main_input)

lstm_out = LSTM( 32 )(x)
```

So the main headline input is supposed to receive sequences consisted of hundred integers. It should be noted that any layer can be named by passing to it a name argument. The main input will be equal to the input shape. This layer which will be embedded can encode the initial input sequence into another sequence consisted of dense five hundred and twelve-dimensional vectors. Finally, a LSTM will transform that vector sequence which will be previously created, into a single vector which will be containing data about the overall sequence.

At this point, the auxiliary loss should be inserted, which will allow newly created embedding and LSTM layer to be trained even in the case if the main loss is much higher positioned in the model. So it would look like the following:

```
auxiliary_output = Dense( 1 , activation= ‘sigmoid’ , name=
‘aux_output’ )(lstm_out)
```

The next step is to auxiliary input data feed into our model by concentrating on the LSTM output previously generated. So auxiliary input will be equal to input containing shape and name which are already generated. A deep densely network will be stacked on top. Finally, the main regression logistic layer will be added. This will define our model, as a model containing two outputs and two inputs.

```
model = Model(inputs=[main_input, auxiliary_input], outputs=
[main_output, auxiliary_output])
```

The final step is to compile our model and assign a certain weight to the auxiliary loss, so we will assign a weight of 0.2 and specify different losses and loss weights for each of these two possible outputs. A dictionary or a list can be used. In this case, the same loss can be used on both of outputs.

```
model.compile(optimizer= ‘rmsprop’ , loss= ‘binary_crossentropy’
,
          loss_weights=[ 1. , 0.2 ])
```

Also, the model can be trained by passing to it lists of target and input arrays. If your model is named by passing to it a name argument, compiled model can be created.

```
model.compile(optimizer= ‘rmsprop’ ,
          loss={ ‘main_output’ : ‘binary_crossentropy’ ,
‘aux_output’ : ‘binary_crossentropy’ },
          loss_weights={ ‘main_output’ : 1. , ‘aux_output’ : 0.2 })


model.fit({ ‘main_input’ : headline_data, ‘aux_input’ :
additional_data},
        { ‘main_output’ : labels, ‘aux_output’ : labels},
        epochs= 50 , batch_size= 32 )
```

Models that use Shared Layers

Another great usage of the functional API, are models which use shared layers. For example, if we consider any dataset and let’s say a dataset of comments on some news headline which we previously discussed. Our model is supposed to tell whether two comments are form the same individual or not. This is also allowing us to compare different users by the similarity of their comments.

The first way to manage this is to create a model which will encode two comments into two vectors, and the logistic regression will be added on top. The probability of two comments being shared by the same person can be calculated. Later the model will be trained on negative comment and positive comment pairs.

Since this problem is symmetric, the mechanism of encoding the initial comment have to be reused both its model structure and weight. Therefore, the second comment can be encoded. To encode the comments, a shared LSTM is used. In order to build the functional API, the comment input will be a binary matrix of certain shape and sequence of hundred and forty vectors.

```
import keras
```

```
from keras.layers import Input, LSTM, Dense
from keras.models import Model
```

Vectors will also be encoded with certain size, and each dimension in the two hundred and six dimensional vector will encode the absence or presence of a character. It will look like this :

From keras to import layers

Input, LSTM, Dense

From keras to import models

Comment a is equall to input

Comment b is equall to input

In order to share any layer across various inputs, the layer has to be instantiated once so later many inputs can be called. Therefore, layers can take certain inputs and return to the vector. It should be noted when the same layers is reused for numerous times, the weight of that certain layer is also being reused, so it is in fact the same layer. Also two vectors can be concatenated using merged layers argumentation.

```
shared_lstm = LSTM( 64 )

encoded_a = shared_lstm(tweet_a)
encoded_b = shared_lstm(tweet_b)

merged_vector = keras.layers.concatenate([encoded_a, encoded_b],
axis=- 1 )
```

The next step is to add on top a logistic regression which will create predictions that are equall to dense. If we want to train the model and to link comment inputs to the certain predictions, model will be equall to inputs of both a and c comments and outputs are our predictions. Use argumentations model.compile and model.fit to configure the model. It will look like the following :

```
predictions = Dense( 1 , activation= ‘sigmoid’ )(merged_vector)

model = Model(inputs=[tweet_a, tweet_b], outputs=predictions)

model.compile(optimizer= ‘rmsprop’ ,
          loss= ‘binary_crossentropy’ ,
          metrics=[ ‘accuracy’ ])
model.fit([data_a, data_b], labels, epochs= 10 )
```

Model is equall to inputs of comment a and comment b

Model outputs are equall to predictions

The Notion of Layer Node

Everytime you call a layer on any input, you are in fact creating a new tensor, in other words you are creating a new output of that layer. Therefore, you are adding a certain node to that layer, connecting an output tensor to some other output tensor. In the case when you call the same layer for numerous times, that layer will own multiple nodes which will be indexed with value of consequtive numbers like zero, one, two, etc.

You may be familiar with previous versions of Keras where you were able to obtain a output tensor of any layer using layer.get output argument, as well as its output shape using layer.output shape. In the case when a layer is linked to multiple nodes, there still won’t be any confusion since the .output argument will return back the certain output of the layer.

So in the case when a single layer has a numerous outputs, the argumentation get output at should be used instead of layer output which is used when there are no multiple outputs. Also for properties like output shape and input shape, the same is true in cases when a layer contains a single node , or if all nodes have a same output and input shape, the argumentation input shape and layer output can be used to define the model. That one shape can be returned using layer.output shape as well as layer.input shape.

```
a = Input(shape=( 140 , 256 ))


lstm = LSTM( 32 )
encoded_a = lstm(a)


assert lstm.output == encoded_a
```

When it comes to the other Keras models, they include models like inception module, convolutional layer or residual connection models, shared vision models, visual question answering models and video question answering models. Shared vision model is a model which re-uses a same processing model for images on two different inputs in order to classify whether two digits are same or different digits.

Keras inception model

```
from keras.layers import Conv2D, MaxPooling2D, Input


input_img = Input(shape=( 3 , 256 , 256 ))


tower_1 = Conv2D( 64 , ( 1 , 1 ), padding= 'same' , activation=
'relu' )(input_img)
```

```
tower_1 = Conv2D( 64 , ( 3 , 3 ), padding= ‘same’ , activation=
‘relu’ )(tower_1)


tower_2 = Conv2D( 64 , ( 1 , 1 ), padding= ‘same’ , activation=
‘relu’ )(input_img)
tower_2 = Conv2D( 64 , ( 5 , 5 ), padding= ‘same’ , activation=
‘relu’ )(tower_2)


tower_3 = MaxPooling2D(( 3 , 3 ), strides=( 1 , 1 ), padding=
‘same’ )(input_img)
tower_3 = Conv2D( 64 , ( 1 , 1 ), padding= ‘same’ , activation=
‘relu’ )(tower_3)


output = keras.layers.concatenate([tower_1, tower_2, tower_3],
axis= 1 )
```

Visual question answering model is a model which can select a correct word answer when question is asked in a natural-language about pictures. This model works by encoding questions into vectors, and then encoding images into vectors. These two are later concatenated, and training takes place and on top is placed a logistic regression over vocabulary of answers which may happen.

Keras vision model

```
from keras.layers import Conv2D, MaxPooling2D, Input, Dense,
Flatten
from keras.models import Model


digit_input = Input(shape=( 1 , 27 , 27 ))
x = Conv2D( 64 , ( 3 , 3 ))(digit_input)
x = Conv2D( 64 , ( 3 , 3 ))(x)
x = MaxPooling2D(( 2 , 2 ))(x)
out = Flatten()(x)
```

```
vision_model = Model(digit_input, out)

digit_a = Input(shape=( 1 , 27 , 27 ))
digit_b = Input(shape=( 1 , 27 , 27 ))

out_a = vision_model(digit_a)
out_b = vision_model(digit_b)

concatenated = keras.layers.concatenate([out_a, out_b])
out = Dense( 1 , activation= 'sigmoid' )(concatenated)

classification_model = Model([digit_a, digit_b], out)
```

Video question answering model is a model which can turn images from question answering model into certain video question answering models. After some proper training, you will be able to create a short video and ask natural language questions about that video. Other Keras models are explained in the next chapters of the book.

Keras video question answering model

```
from keras.layers import TimeDistributed

video_input = Input(shape=( 100 , 3 , 224 , 224 ))
encoded_frame_sequence = TimeDistributed(vision_model)
(video_input)  # the output will be a sequence of vectors
encoded_video = LSTM( 256 )(encoded_frame_sequence)  # the
output will be a vector
```

```
question_encoder = Model(inputs=question_input,
outputs=encoded_question)

video_question_input = Input(shape=( 100 ,), dtype= ‘int32’ )
encoded_video_question =
question_encoder(video_question_input)

merged = keras.layers.concatenate([encoded_video,
encoded_video_question])
output = Dense( 1000 , activation= ‘softmax’ )(merged)
video_qa_model = Model(inputs=[video_input,
video_question_input], outputs=output)
```

Chapter 3 Keras Layers

All Keras layers have some methods in common. For example, argumentation layer gets weights will return the layer's weights as a listing of arrays. Another way around, argumentation layer will set weights of the layer's weight from a listing of Numpy arrays together with the same output's shapes. Argumentation layer that gets configuration will return a dictionary which is consisted of a configuration of any layer. Later that layer can be instantiated from configuration using argumentation layer configuration.

```
layer = Dense( 32 )
config = layer.get_config()
reconstructed_layer = Dense.from_config(config)
```

If a layer consists of a single node and if it's not a connected or shared layer, you can easily get output and input tensor, as well as output and input shape using argumentations layer input, layer output, layer input shape and layer output shape.

```
from keras import layers

config = layer.get_config()
layer = layers.deserialize({ 'class_name' :
layer.__class__.__name__,
                         'config' : config})
```

However, if the layer is consisted of multiple nodes as well as of connected layers, you should use argumentations layer get input at the indexed node, the layer gets output at an indexed node and same for the getting input and output shape, using node index.

Core Keras Layers

Core Keras layers are densely connected layers. To implement an overall operation, dense argumentation must be used. Therefore, an output will be equal to activation. Here the activation is the function which will pass as the argument. For example, classifying an initial layer of any sequential model will be by adding dense argumentation. The model will later take its input arrays from its shape. The output of the model will also be taken as arrays of its shape. After you specified the initial layer, there is no need no specify the following layers.

```
keras.layers.core.Dense(units, activation= None , use_bias= True ,
kernel_initializer= ‘glorot_uniform’ , bias_initializer= ‘zeros’ ,
kernel_regularizer= None , bias_regularizer= None ,
activity_regularizer= None , kernel_constraint= None ,
bias_constraint= None )
```

Arguments for core Keras layers include various methods. Units are dimensionality of an output space consisted of positive integers. The activation function is used to specify inputs and outputs, and if you don’t specify activation function, there will be no activation applied.

```
model = Sequential()
model.add(Dense( 32 , input_shape=( 16 ,)))

model.add(Dense( 32 ))
```

Use bias is an argument which can indicate if a layer is using a bias vector. Kernel initializer is used to initiate kernel weights. Bias initializer is used to initialize bias vectors. Kernel and bias regularizer arguments are used to regularize functions applied to kernel weight and bias vectors. Activity regularizer is argumentation used to regularize any functions which are applied to an output of a layer. Kernel and bias constraints are argumentations which are used to constraint functions which are applied to kernel weights and bias vectors.

Activation

```
keras.layers.core.Activation(activation)
```

When it comes to the usage of these functions, activations can be used both through the activation argument as mentioned previously and through an activation layer. Available activations are softmax where softmax is equal to x tensor and axis which is an integer. The softmax normalization can be applied along with axis value. An output of any softmax transformation is the tensor.

Elu and Selu are also available activation functions working similarly as softmax. Other activations include tanh, hard sigmoid, linear, softsign and softplus. All of these work in pretty much the same way. Advanced activations include eakyRelu and Prelu, and these are more complex than these simple Theano and Tensorflow functions.

When it comes to the initializers in Kernel, they are used to define certain ways of setting random weights of layers. The keyword arguments are already familiar since those are mentioned previously. Initializers which are available include zeros which can generate tensors that are initialized to zero. Other initializers are which can generate tensors that are initialized to one, and constant initializer can generate tensors which are initialized to any constant value.

Random and normal initializers can generate tensors which have a normal distribution. The random uniform initializer can generate tensors that have a uniform distribution.

Truncated normal initializer can generate a normal truncated distribution and values which are generated are similar to those generated from a random and normal initialization. This is the most commonly used initializer for generating neural filters and weights.

Variance scaling is another initializer which is capable adapting scales to their weights. The orthogonal initializer can generate any orthogonal matrix. Identity is another initializer which can generate any identity matrix but is only used for dimensional matrices. Lecun uniform initializer draws various samples from a previous uniform distribution within certain limits.

Glorot normal is another initializer which can draw samples but this time from normal truncated distribution which is centered on zero. In this initialization, fan in is a function which indicated some input units in a weight tensor and fan out is another way around, indicating some output components in a weight tensor.

On the other hand, regularizers let you apply penalties to any layer activity or parameters during the optimization process. Therefore, these applied penalties will be incorporated in loss functions which are optimized for the network. The penalties can be applied one per a single layer basis.

The functional API depends on a layer, but layer like dense or conventional one dimension have a united API. Three keywords are exposed with these layers, and those are kernel, activity and bias regularizers. Any function which is consisted of a matrix weight and later returns to a loss tensor can be used in this method as the regularizer.

Allowing setting certain constraints like non-negativity, constraint functions are used on certain network parameters during the optimization process. These constraints can be applied to a single layer. The exact functional API depends on a layer just like when regularizers are used.

These layers can expose two keyword arguments, and those are bias and kernel constraints. Constraints which are available are maximum norm constraint, non-negativity constraint and unit-form constraint that enforces a matrix to have a certain unit norm along with the final axis.

When it comes to the input and out shapes, the most common situation is two-dimensional input and shape using argumentations batch size and input dim. In this case, the output will have the shape the size of the batch and its components. Dropout function applies dropout to any input.

It consists of the fraction which is a rate of input components to zero at every update during certain training time. This can help to prevent overfitting. Arguments used in dropout functions are the rate which is the fraction of input between zero and one. Noise shape is one dimension integer tension which is representing a shape of a binary dropout which can later be multiplied with an input. Argument seed is a Python integer which is used as a random seed.

Dropout

```
keras.layers.core.Dropout(rate, noise_shape= None , seed= None )
```

Flatten is used to flatten an input without causing any effects to a batch size. Reshape is used to reshape an output to a particular shape. The argument is target shape which is the tuple of integers, and a batch axis is not included. Permute function can permute dimensions of input by a certain pattern.

Flatten

```
keras.layers.core.Flatten()
```

This is very useful when it comes to the linking RNN'S and convents. Dims is used argument where indexing starts at one.

However, sample's dimensions are not included. With permute function, values of an input are permuted.

Permute

```
keras.layers.core.Permute(dims)
```

Repeat vector function can repeat an in input numerous times. A single argument is used which is n that is repetition factor and can be any integer. Lambda function can wrap arbitrary expression as any layer object.

Arguments which are used include output shape which can generate an expected output shape from any function. Input tensor is taken as the first argument.

Permute Example

```
model = Sequential()
model.add(Permute(( 2 , 1 ), input_shape=( 10 , 64 )))
```

Lambda function and output shape argumentation are only relevant when Theano is used, and it can be both tuple and function. If it's a tuple, it can only specify an initial dimension onward. On the other hand a sample dimension is simply assumed to be the same as an input or output.

Lambda

```
keras.layers.core.Lambda(function, output_shape= None , mask=
None , arguments= None )
```

Activity regulation is another Keras layer which can apply an update to any cost function based on its input activity. Arguments which are used are L1 and L2 which are both regularization factors of the positive float. Masking is also the core layer of Kears which can mask a sequence by using certain mask value to skip timesteps.

Activity regularization

keras.layers.core.ActivityRegularization(l1= 0.0 , l2= 0.0)

For every timestamp in an input tensor or the first dimension of a tensor, if all values in an input tensor are equal to certain mask value, this timestep can be masked or skipped in all following and downstream layers. The only requirement for doing so is that they support masking layers.

Masking

keras.layers.core.Masking(mask_value= 0.0)

Convolutional Keras Layers

Temporal convolution or one-dimensional convolution layer is denoted as Conv1D. This is the layer which creates a convolution kernel which is convolved with an input layer over a single or temporal dimension. By creating this layer tensor of outputs is produced. Bias vector can be created use bias function is true then this vector is added to outputs. In the case when activation is none, it is also applied to all outputs.

keras.layers.convolutional.Conv1D(filters, kernel_size, strides= 1 , padding= ‘valid’ , dilation_rate= 1 , activation= None , use_bias= True , kernel_initializer= ‘glorot_uniform’ , bias_initializer= ‘zeros’ , kernel_regularizer= None , bias_regularizer= None , activity_regularizer= None , kernel_constraint= None , bias_constraint= None)

When you use convolution layers as an initial layer of your model, input shape argument should be added which would consist of integers or tuples. Integers will be none if the sequence of ten vectors consisted of one hundred twenty-eight dimensional vectors, or input shape may consist of none for sequences which vary in length.

Arguments which are included in any convolution model are filters, strides, dilation rate, etc. Filters are the integer which refers to the dimensionality of an output space. In other words, it is a number of output filters in any convolution. Kernel size is another argument common with convolution models. Kernel size is a tuple or integer which is specifying a length of stride of that certain convolution. This is compatible with specifying dilation rate and both are the same value of one.

```
keras.layers.convolutional.Conv2D(filters, kernel_size, strides=( 1 ,
1 ), padding= ‘valid’ , data_format= None , dilation_rate=( 1 , 1 ),
activation= None , use_bias= True , kernel_initializer=
‘glorot_uniform’ , bias_initializer= ‘zeros’ , kernel_regularizer=
None , bias_regularizer= None , activity_regularizer= None ,
kernel_constraint= None , bias_constraint= None )
```

Padding argument refers to one valid, one same or one causal result. Valid in this case means that there is no any padding and the same padding is, in fact, a case-sensitive padding. Their results of any padding include inputs like such as a certain output that consists of the same length as the initial input.

```
keras.layers.convolutional.ZeroPadding1D(padding= 1 )
```

Causal padding results in dilated or causal convolutions which may be the result when output is not dependable on input. Padding pattern may be very useful when it comes to the creating model of temporal data when the model isn’t supposed to violate the order, in this case, temporal order.

```
keras.layers.convolutional.ZeroPadding2D(padding=( 1 , 1 ),
data_format= None )
```

Dilated rate argument is the tuple or integer list or a single integer or tuple. This argument specifies a dilation rate which is used for dilated convolution. As mentioned previously, dilated rate value is one which is the same as strides value which is also one. Activation argument is the function which can be used. No activation will be applied if you don't specify anything. This is a linear activation as shown below.

$$a\,(x) = x$$

Argument use bias is the function when any layer uses a bias vector. Kernel initializer is used for any kernel matrix weights. The similar argument is bias initializer which is used for any bias vector. Kernel and bias regularizer arguments are functions which are applied to kernel matrix weights and bias vectors. Activity regularizer is applied to outputs of layers. Kernel and bias constraints are constraint functions applied to kernel matrices and bias vectors.

The input shape of convolution model which has three-dimensional tensor consists of batch size, steps and input dim. On the other hand, the output share of same three-dimensional tensor consists of batch size, new steps, and filters. This step value can be changed due to strides or padding.

```
keras.layers.convolutional.Conv3D(filters, kernel_size, strides=( 1 , 1 , 1 ), padding= 'valid' , data_format= None , dilation_rate=( 1 , 1 , 1 ), activation= None , use_bias= True , kernel_initializer= 'glorot_uniform' , bias_initializer= 'zeros' , kernel_regularizer= None , bias_regularizer= None , activity_regularizer= None , kernel_constraint= None , bias_constraint= None )
```

Another convolution layers that are two-dimensional convolution layers are denoted as Conv2D. The example of this layer is the spatial convolution of images. This layer can create a convolution kernel which is convolved with any input layer in order to produce a tensor of different outputs. If use bias function is true, a bias vector will be created and added to numerous outputs. The same thing as in Conv1D, if there is no activation applied, the function will be applied to every output as well.

If you are using this two-dimensional layer as your initial layer of you model, keyword argument of input shape must be provided. Therefore, the input shape will be integer or tuple without the sample axis. Same arguments as in one-dimensional convolution layer are used, but with few additions. Data format argument is the function which can arrange the dimensions in any output.

Functions which are used are channels last and channels first which are strings between inputs. Therefore, this channels last arrangement of input dimensions is in correspondence to inputs with certain shapes which include batch, height, width, and channels. Channels first are on the other hand in correspondence to inputs consisted of batch, channels, height, and width.

Channels last argument correspond to inputs with

batch, height, width and channels shape

Channels first argument corresponds to inputs with

batch, channels, height, and width shape.

Input shapes for four-dimensional tensor will consist of sample, channels, rows and cols shape. Data format will be equal to channels first. Output shape of four-dimensional tensor will consist of samples, filters, new rows and new cols shape. Data format will also equal to channels first or to four-dimensional tensor share consisted of the sample, new row, new cols, and filters. Cols and Rows values may change due to padding.

Another type of convolution layers is separable two-dimensional convolution denotes as SeparableConv2D. These convolutions include first performing a spatial depth-wise convolution. This spatial convolution is affecting each input channel independently.

Following is a pointwise convolution that mixes output's channeling results.

```
keras.layers.convolutional.SeparableConv2D(filters, kernel_size,
strides=( 1 , 1 ), padding= 'valid' , data_format= None ,
depth_multiplier= 1 , activation= None , use_bias= True ,
depthwise_initializer= 'glorot_uniform' , pointwise_initializer=
'glorot_uniform' , bias_initializer= 'zeros' ,
depthwise_regularizer= None , pointwise_regularizer= None ,
bias_regularizer= None , activity_regularizer= None ,
depthwise_constraint= None , pointwise_constraint= None ,
bias_constraint= None )
```

If you want to control how many channels are generated from the output per single input channels, you use a depth multiplier argument. For better understanding think of these separable convolutions as a certain way to factorize any convolution kernel into smaller kernels.

There are few additional arguments used in separate convolution layers. First is data multiplier which we already mentioned. Depth-wise and pointwise initializers are functions for depth-wise and pointwise kernels matrices. Depth-wise and pointwise constrain arguments are constraint functions which are applied to depth-wise and pointwise kernel matrices.

Input shape of four-dimensional tensor will consist of the batch, filters, new rows and new cols input shape. The data format will be equal to channels first. Output shape of four-dimensional tensor will consist of the batch, filters, new rows and new cols input shape. Data format will equal to channels first just as in input shape. Cols and Rows values may change due to padding.

Transposed convolution layer is another type fo convolution layers. It is denoted as Conv2DTranspose and often called as Deconvolution. There was a need for these layers due to desire to use a transformation which would be going in the opposite direction.

```
keras.layers.convolutional.Conv2DTranspose(filters, kernel_size,
strides=( 1 , 1 ), padding= ‘valid’ , data_format= None , activation=
None , use_bias= True , kernel_initializer= ‘glorot_uniform’ ,
bias_initializer= ‘zeros’ , kernel_regularizer= None ,
bias_regularizer= None , activity_regularizer= None ,
kernel_constraint= None , bias_constraint= None )
```

For a better understanding of transposed convolution, imagine from something which has the certain output shape of some convolution to something else which has a shape of its input. So, while maintaining a connection pattern, this would be compatible with transposed layer of deconvolution.

If you use this transposed layer as the initial layer of your model, you should provide the key argument of input shape which doesn’t include the sample axis. Data format will be equal to channels last in this case as well. Arguments which we previously discussed are used in transposed convolution layer as in previous convolution layers.

Input shape of four-dimensional tensor will consist of batch, channels, rows and cols shape and data format will be equal to channels first. Output shape of four-dimensional tensor will consist of the batch, filters, new rows and cols shape and data format will be equal to channels last. Here cols and rows may change their values due to padding.

4D tensor shape

Batch, filters, new rows and new cols

Another type of convolution layer if three-dimensional convolution layer denoted as Conv3D. This layer is often used as the spatial convolution of volumes. This layer can create a certain convolution kernel that is convolved with any layer's input to produce a particular tensor of different outputs. If you use argument use bias, a bias vector will be created and added to every output. If there is no activation applied, it will be applied to every output as well.

The key argument for this layer if uses as an initial layer of the model are input shape. Input shape doesn't include the sample axis and consist of integers. Input shape of five-dimensional tensor will consist of samples, channels, convolution dim1, convolution dim2 and convolution dim3. Output shape of five-dimensional tensor will consist of samples, filters, new convolution dim1, new convolution dim2 and new convolution dim3. Convolution values can be changed due to padding.

5D tensor shape

Sample, channels, convolution dim1, convolution dim2, convolution dim3

Cropping convolution one-dimensional and two-dimensional layers can crop along any time dimension in the model. Examples are cropping of images and temporal sequences. Input shape of three-dimensional tensor consists of batch, axis which is supposed to be cropped and features. Output shape of three-dimensional tensor consists of the batch, already cropped axis and features.

```
keras.layers.convolutional.Cropping1D(cropping=( 1 , 1 ))
```

Two-dimensional dropping layers are most common for images, where it crops along dimensions of any image, its height, and width. Arguments include cropping and data format. Input shape of the four-dimensional tensor is data format equal to channels last. Output shape of such a tensor is data format equal to channels last, so input shapes include batch, channels, chopped raws, and cols.

Cropping three-dimensional layer is used as spatiotemporal or spatial.Arguments include cropping and data format. Input shape of five-dimensional tensor will be data format equal to channels first. The shape will consist of the batch, first axis which will be cropped, the second axis which will be cropped, third axis to be cropped and depth.

```
keras.layers.convolutional.Cropping3D(cropping=(( 1 , 1 ), ( 1 , 1 ),
( 1 , 1 )), data_format= None )
```

Output shape of five-dimensional tensor will include batch, the first axis which is cropped, the second axis which is cropped, third axis which is cropped and depth.

5D tensor shape

Batch, first axis to be cropped, second axis to be cropped, third axis to be

cropped and depth

There is also available convolution layer upsampling one dimensional. This layer repeats temporal step size along the time axis. Input and output shape of three-dimensional tensor include batch, steps, unsampled steps, and features. There is also available up sampling layer for two-dimensional inputs. This layer repeats columns and rows of the data by size 1 or size 0 respectively. Arguments include size and data format.

```
keras.layers.convolutional.UpSampling1D(size= 2 )
```

Convolution layer up sampling three-dimensional, repeats all three dimensions of the data by size 2, size 1 and size 0. Input and output shapes of five-dimensional tensor include batch, dim 1, dim2, dim3, channels for input shape and batch, unsampled dim1, unsampled dim2, unsampled dim3 for output shape.

The last convolution layer available in Keras is the zero-padding layer which includes layers for one dimensional, two dimensional and three-dimensional inputs. Input shape for three-dimensional tensor includes batch, axis which will be padded and features. Output shape includes batch, already padded axis and features.

```
keras.layers.convolutional.ZeroPadding1D(padding= 1 )
```

Recurrent Keras Layers

Recurrent keras layers use its classes like GRU, Simple RNN and LSTM. This layer follows the certain specifications of its classes and accepts arguments weights, which is a listing of Numpy array that is set as starting weight. This Numpy listing should contain three elements, input dim and output dim, output dim and output dim and just output dim.

```
keras.layers.recurrent.Recurrent(return_sequences= False ,
return_state= False , go_backwards= False , stateful= False ,
unroll= False , implementation= 0 )
```

Argument return sequences can return to the full sequence or last output in any output. Return state argument can return to the last state with the addition of returning to the output. Go backward argument if is true, can return the process of input sequences backward and to reversed sequences.

The stateful argument if is true, the last state of any sample at its index can be used in any batch as a starting state for that sample in all following batches.

```
model = Sequential()
model.add(LSTM( 32 , input_shape=( 10 , 64 )))
model.add(LSTM( 16 ))
```

```
model = Sequential()
```

```
model.add(LSTM( 64 , input_dim= 64 , input_length= 10 ,
return_sequences= True ))
model.add(LSTM( 32 , return_sequences= True ))
model.add(LSTM( 10 ))
```

Unroll argument if is true the overall network can be unrolled, and the symbolic loop will be used instead. This argument is only proper for shorter sequences, and it can speed up an RNN, but it is more memory-intensive.

Implementation argument if is set to zero, the certain RNN can use an implementation which uses a few, but larger matrices, which is running faster on desktop computers but also it is consuming more memory.

If you set it to one, the RNN this time will use smaller but more matrix products. If you set it to two, the RNN will gather the forget gate, input gate and output gate into one matrix, and more time-efficient parallelization will be enabled.

Input dim argument creates dimensionality of any input integer. This argument is mandatory when you are using this layer as an initial layer of your model. Input length argument creates the certain length of any input sequence to specify when the sequence is constant. This argument is mandatory when it comes to the connecting dense and flatten layers upstream.

Input shapes of three-dimensional tensor will consist of batch size, timesteps and input dim. There are also optional two-dimensional tensor consisted of batch size and output dim. Output shape uses argument return state and returns sequences.

If you use argument return state, the initial tensor will be output, while the rest of tensors will be last states and each of tensors will consist of batch size and units shape. If you use argument return sequences, the three-dimensional tensor shape will consist of batch size, timesteps, and units.

3D tensor shape

Batch size, timesteps and input dim

When it comes to the masking, it is possible in this layer for any input data consisted of a variable number of timesteps. In order to mask your layers, you should use embedding layer and set mask zero argument parameter to be true.

Embedding Keras Layers

Embedding Keras layers can be used only as the initial layer in a model. The model will be sequential with certain input length. The input of the model will be an integer regarding matrices of sizes batch and input length.

Arguments include input dim which is an integer that is greater than zero and that refers to the size of certain vocabulary and the maximum integer index can be plus one. Output dim is the integer greater than zero which is the dimension of dense embedding.

```
keras.layers.embeddings.Embedding(input_dim, output_dim,
embeddings_initializer= 'uniform' , embeddings_regularizer=
None , activity_regularizer= None , embeddings_constraint= None
, mask_zero= False , input_length= None )
```

Embedding initializer and embedding regularizer are functions which initialize and regularize embedding matrices. Embedding constraint is the argument which applies constraint function to embedding matrices. Mask zero argument is sort of special padding value which should be masked.

This comes to be very useful in recurrent layers as well, where variable input length could be taken. It it is set to be true then all layers which are subsequent will need to support masking, or there will be the exception raised. If you set mask zero arguments to be true, the consequence will be the index of value zero which cannot be used in vocabulary.

Instead, input dim should be the same value of vocabulary plus one.Input length argument will list the length of all input sequences if it is a constant. This argument is mandatory when it comes to the linking dense and flatten layers in the upstream direction. Without input length, the shape of any dense outputs will not be computed.

Example

```
model = Sequential()
 model.add(Embedding( 1000 , 64 , input_length= 10 ))

 input_array = np.random.randint( 1000 , size=( 32 , 10 ))

 model.compile( ‘rmsprop’ , ‘mse’ )
 output_array = model.predict(input_array)
 assert output_array.shape == ( 32 , 10 , 64 )
```

2D tensor shape

Batch size and sequence length

Chapter 4 Deep Learning Algorithms

Deep learning is as we already mentioned, about learning numerous levels of abstraction and representation which can help make sense of any data such as text, sound, and images. Deep learning is an area of Machine Learning research that has been introduced with the intention of moving Machine Learning techniques closer to Artificial Intelligence. There are completely supervised learning algorithms, semi-supervised algorithms, and unsupervised learning algorithms.

Supervised Learning Algorithms

Logistic Regression

Logistic regression is entirely probabilistic and linear classification. Its parameters include the weight of matrix denoted as W and detonated with b a bias vector. To classify it, initial vectors must be implemented into a collection of hyperplanes, and each of these hyperplanes will be in correspondence with a certain class.

Logistic regression in Keras

```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

The distance between the input to this hyperplane will be reflecting the probability of the input to be a component of a corresponding class. Therefore, the logistic regression model is a model of prediction.

In order to code this in Keras, you should start with value zero which would be the matrix of certain shapes. After initializing the weight, you should initialize bias vector as well. It should be noted that the parameters of your model must stay at a persistent state before and after training.

Reshaping inputs

```
input_dim = 784 #28*28
X_train = X_train.reshape(60000, input_dim)
X_test = X_test.reshape(10000, input_dim)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

Therefore, both variables bias vector and weight of matrix will be referred to as certain symbols. After initializing those values, the model must be configured.Softmax activation argument and the dot are used to compute the probability vector.

Probability vector

P (yIx, W, b)

The result of the probability vector will be the y value given the value x which is a symbolic variable of this type of vector. In order to learn parameters from an optimal model, an involvement of loss function and its minimizing is mandatory. It is common in these situations to use negative probability as the loss.

The next step is to define a logistic regression created class, that will show the fundamental behavior of logistic regression class. In order to start, symbolic variables must be allocated for training inputs and to their corresponding classes. This is often used when you when it comes to the linking instances of any classes to develop a deep network.

Converting class

```
from keras.utils import np_utils
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

The output of any layer can pass as an input to the previous layer positioned above. Using instance method, cost variable is minimized.

Building the model

```
from keras.models import Sequential
from keras.layers import Dense, Activation
output_dim = nb_classes = 10
model = Sequential()
model.add(Dense(output_dim, input_dim=input_dim,
activation='softmax'))
batch_size = 128
nb_epoch = 20
```

In order to learn created model in Python, you should manually derive expressions for a gradient of a loss corresponding to certain parameters which you created. In Keras this is very simple since it performs differentiation automatically and applies particular math transformations which improve overall numerical stability.

Compiling the model

```
model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X_train, Y_train, batch_size=batch_size,
nb_epoch=nb_epoch,verbose=1, validation_data=(X_test, Y_test))
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

When it comes to the testing of created model, the number of misclassified examples is what we are interested in and not only in the probability. Test model argument must be applied, and after a validate model argument will retrieve the value.

The difference between these two arguments is that test model draws its batches from a testing collection, whole validate model draws its from a validation collection.

Saving model and weights

```
json_string = model.to_json() # as json
open(‘mnist_Logistic_model.json’, ‘w’).write(json_string)
yaml_string = model.to_yaml() #as yaml
open(‘mnist_Logistic_model.yaml’, ‘w’).write(yaml_string)
```

```
model.save_weights(‘mnist_Logistic_wts.h5’)
```

```
model = model_from_json(open(‘my_model_architecture.json
```

```
model = model_from_yaml(open(‘my_model_architecture.yaml
model.load_weights(‘my_model_weights.h5’)
```

Multi-layer Perception

You are already familiar with the fact that main focus is a model. To create a multi-layer perception, the simplest sequential model will be used with a stack of subsequent layers. You can also add more layers in the arrangement of the computation you are going to perform. Your starting layer of your model must contain specified shape of the input. You will define it using input dim argument, and it will consist of a certain number of input features.

Creating a sequential model

```
from keras.models import Sequential
```

```
model = Sequential(…)
```

The argument will be expecting an integer. Layer models may have few features in common including their weight initialization and activation functions. Unit argument will specify which type of initialization will be used for every layer if you specify it.

Common types of layer initializer are zero, normal and uniform. You should specify the kind of activation you will use by applying activation argument to layers. There are numerous layer types which can be used such as:

- *Dense layer: fully linked layer that is the most used on multi-layer models*
- *Dropout layer: applying dropout to the created model and sets a fraction of the input to value of zero*
- *Merge layer: combine the inputs from numerous models into one model*

After you define your model, it has to be compiled. This will create structures which are used by the underlying backend like TensorFlow and Theano which will execute created model during training. Compile your model using three features model optimizer, metrics, and the loss function.

Model compilation

```
model.compile(optimizer=, loss=, metrics=)
```

The model optimizer is a search method which is used to update weights features in your model. The next step is to create an optimizer which will be passed to its compile function by the optimizer argument. This will allow you to perform configuration of the optimization using it's arguments. The most commonly used gradient optimizers which can add various parameters to the optimizer are:

- *SGD: stochastic gradient optimizer*

- *RMSprop: optimization method for learning adaptive rate*
- *Adam:adaptive moment estimation*

Model optimizers

```
sgd = SGD(…)
model.compile(optimizer=sgd)
```

Objective function or loss function will be used to compile function using the loss argument. Commonly used loss function arguments are MSE or in other words Mean Squared Deviation, categorical cross-entropy and binary cross-entropy used for binary and multi-class logarithmic loss. Metric parameters will be evaluated by the model itself during model training.

The model will be trained on Numpy arrays, and you should use fit function argument. Training will both specified batch size and epochs. Epochs will be the number of times in which created model is exposed to a training dataset.

Model training

```
model.fit(X, y, epochs=, batch_size=)
```

Batch size will be the number of instances of training which will be shown to created model before any updating of the weight is performed. The fit function argument will allow you to some fundamental evaluation of your model during model training. Once you are finished with your model training, you can use created the model to make predictions on various test data.

You can use different methods to evaluate your model, and those are as following:

- *Model evaluate: calculates the loss value for any input data*

- *Model predicts: generates network output for any input data*
- *Model predict classes: generates class outputs for any input data*
- *Model predicts probability: generated class probabilities for any input data*

Summarizing your model

model.summary()

Creating an image of the model

from keras.utils.visualize_util import plot

plot(model, to_file='model.png')

Deep Convolutional Network

In a deep convolutional network created with Keras a created model can identify objects like images. This may come as a difficult task to automatically identify objects in various images, since there are an infinite number of positions of objects, permutations, lighting, etc.

Therefore, a huge deep learning dataset was developed by the CIFAR, which is Canadian Institute for Advanced Research. Their dataset consists of sixty thousand images which are divided into ten classes.

This dataset is regularly split, and fifty thousand images are used for training of models, and the remaining images are used for evaluating model's performance.

In order to create a deep convolutional network in Keras, first, the CIFAR-10 dataset must be loaded into Keras. Keras can automatically download these standard datasets just like CIFAR-10, and it will be stored in the Keras directory. It may take some time to download it since the dataset is large at hundred and sixty-three megabytes.

Once you download it, it will be ready for use. You will see that each downloaded image is represented as a three-dimensional matrix and its dimensions are blue, red and green height and width. In order to plot images run matplotlib.

```
from __future__ import print_function

import keras
from keras.datasets import cifar10
from keras.processing.image
import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense
Dropout, Activation, Flatten
From keras.layers imports Conv2D
MaxPooling2D
```

Using a convolutional neural network or simply CNN, the CIFAR-10 problem is solved. In order to start, you have to define function and classes. You can initialize a random number seed with any constant which will ensure that the results are able to reproduce. The next step is to load the dataset downloaded. You will notice that pixel values are between zero to two hundred fifty-five for each of the blue, red and green channels. It should be noted that the data will be loaded as integers. Therefore, they have to be cast to theirs floating particular point values which will enable the division.

The output values will be defined as a vector of integers with values between zero and one for every class. If you want to transform them into for example binary matrix, you should use a hot encoding.

Therefore, we should expect created binary matrix to consist of the width of ten classes since we already know that there are ten classes for this certain problem.

The next step is to define our CNN model and evaluate how well it can perform on the problem. The structure which will be used consists of two layers which are followed by maximum pooling and flattening. The network will consist of fully linked layers which can make predictions. Convolutional neural network structure can be summarized as follows:

1. Setting up of the convolutional initial layer
2. Setting up of the dropout layer
3. Activation function and weight constraint argument
4. Setting up of the max pool layer
5. Connecting layers completely
6. Setting up of the dropout layer
7. Connecting output layer completely with ten units
8. Softmax activation argument

The Semi-supervised and Unsupervised Algorithms

Auto Encoders

Auto-encoding is compression algorithm in which decompression and compression functions include the following:

1. *Data specification*
2. *Lossy*

3. *Functions learned automatically*

Therefore, auto-encoders are always data specified, meaning that they are only able to compress data which is previously trained. Also, auto-encoders are lossy, meaning that outputs which are compressed will be always degraded in comparison to the initial inputs.

```
from keras.layers import Input, Dense
from keras.models import Model
```

Auto-encoders are learned from data examples automatically, meaning that they can be easily trained, and only proper training data is required. In order to build auto-encoder in Keras three things are required a decoding and encoding function and a particular distance function between the amount of data loss among the decompressed and compressed representation.

To start you should connect initial layer fully which will serve as decoder and encoder. The next step is to create independent encoder model as well as independent decoder model. These created models now have to be trained in order to reconstruct digits. Using function binary cross-entropy loss, the model will be configured.

Separate encoder model

```
encoder = Model(input_img, encoded)
```

The next step is to prepare input data and labels which will be discarded since we are only interested in decoding and encoding input data. Values between zero and one should be normalized and data flattened into certain vectors. The next step is to train created model for fifty epochs. This is the basic approach and simplest auto-encoder neural network model.

Configuring your model

```
autoencoder.compile(optimizer=‘adadelta’,
loss=‘binary_crossentropy’)
```

Chapter 5 Applications of Deep Learning Models

At this time it may be hard to find a problem where deep learning models are not used in order to solve it. Today, deep learning models and techniques are involved in a great range of scientific problems and many real-life situations. In this section of the book, we will see what are the most common applications of deep learning models.

Automatic Speech and Image Recognition

The fact is that automatic speech recognition was revolutionized by deep learning models especially with LSTM or long short-term memory recurrent neural networks RNNs. This deep learning model is able to vent gradient problems and is able to learn various deep learning tasks. These tasks involve intervals of every speech event which is separated by numerous discrete time steps, and each step is corresponding to ten ms.

LSTM neural network is ahead of the competition especially ahead of the traditional speech recognizers working on certain tasks. Over a decade ago, LSTM achieved great results on various tasks like discriminative keyword spotting. Its performance has been doubled since in 2015 Google's speech recognition started to use it.

The success in speech recognition was initially based on smaller recognition tasks developed from the TIMIT data collection which is commonly used as data collection for various evaluations. This initial set contained six hundred thirty speakers which were able to read ten sentences. Many configurations were able to be performed.

The TIMIT task was developed for phone sequence recognition which allows weak language models, unlike word sequence recognition. Therefore, components of speech recognition were easily analyzed. Certain analyses on TIMIT around 2010 were the stimulus to invest in deep learning for automatic speech recognition. This led to deep learning models to be dominant in this industry.

Later in 2010, deep learning models from TIMIT data collection were further extended and contained larger data sets. All major speech recognition systems such as Skype, Google Translator, Google Now, Apple Siri, Microsoft Cortana, Amazon Alexa and others are based on deep learning models.

The most common data set for image classification is the MINST which contains handwritten digits and includes sixty thousand training and ten thousand test examples.

Multiple configurations can be tested due to its small size. In 2012 significant impacts in object and image recognition occurred due to the GPU implementation of neural networks. Also, implementations of convolutional neural networks or CNNs served as a great impact in this field.

Image classification was extended to the more challenging problems of generating captions or descriptions for images, most commonly as a combination of LSTMS and CNNs. Since the introduction of deep learning models into automatic image recognition, trained vehicles can interpret three hundred sixty degrees views.

Another great example is FDNA or Facial Dysmorphology Novel Analysis which is used to analyze certain cases of human disfiguration which is connected to a database containing genetic syndromes.

Natural Language Processing

Deep neural network models have been used extensively since the 200s for implementing natural language models. The most commonly are used RNNs or recurrent deep neural networks mostly LSTMs that are the most appropriate for data in sequence arrangement like language. Language modeling and machine translation were improved by usage of deep neural networks LSTMs.

Other important methods and techniques used in this particular field are word embedding and negative sampling. Word embedding in a representational layer in a deep learning model which is able to transform a word into a representation of the word that is positional.

This positional representation of the word is relative to remaining words in the data collection. The position of the word is represented as a point in any vector space of deep learning architecture.

Word embedding created as the recurrent neural network as an initial layer enables the network to divide phrases and sentences using a compositional grammar vector. You can think of this grammar vector as PCFG or probabilistic context free grammar which is implemented by recurrent neural network model.

In natural language processing problems like contextual entity connecting, constituency dividing, information retrieval, machine translation, writing style recognition and sentiment analysis deep neural models have accomplished excellent results.

Another field in which deep learning models are commonly used is toxicology and drug discovery. It is familiar that a great number of drugs fail when it comes to the winning regulatory approval. These failures are mostly due to insufficient on-target effect or other unexpected toxic effects.

A team of researchers in 2012 used a deep neural network in order to predict the biomolecular target of a certain drug. Later in 2014, researchers used deep learning neural network in order to detect the toxic effect and off-target effects on an environmental chemical in food, various households products, and particular drugs.

Researchers improved deep learning neural networks for various drug discovery. They combined data from various sources. For instance AtomWise in 2015 introduced the first learning model which is used for structure-based drug design. This model is called AtomNet and later was used to make various predictions of novel biomolecules for different disease targets like multiple sclerosis and Ebola virus.

Deep learning models are also used in customer relationship management and for various direct marketing settings. Neural networks are used to predict the value of probable direct marketing actions which are defined in RFM variables. The final value is estimated value function which is proven to have an interpretation as customer value.

Recommendation systems also use deep learning models in order to extract significant features for any latent factor in music recommendations. So, deep learning models are applied in order to learn customer preferences from various domains. This model uses content-based and hybrid collaborative approach and enhances recommendation in numerous tasks.

Various autoencoders are used in bioinformatics in order to predict gene ontology and gene-function relationships. Also in medical informatics, neural network models are used in order to predict sleep quality which is based on various data from wearables. Electronic health record model which is based on the deep neural network is used to predict various health complications.

Video Game Development

Various deep learning models can be used in video game development. Most games use integrated deep convolutional network model which is trained in the completely supervised learning context. Most deep learning models for video games problems and tasks must rely on more complex techniques like deep-Q learning. However, the deep supervised model that requires a fewer number of substantial resources and easier training time still can perform well.

This supervised model is able to perform great at human reaction speed in many classic games including Super Smash Bros and others. So learning task must be framed as a thirty class problem, any convolution neural network structure will achieve eighty percent top-1 and ninety-five percent top-3 accuracy from validation.

With some fine tuning, deep learning model can be competitive while live simulating with the highest level of AI that is built into the game. Later, evidence will be shown through network visualizations, and the network will be successfully leveraging information that is temporal while inferencing to aid in whatever decision you make.

Therefore, supervised convolution neural networks can perform great in challenging prediction problems and tasks and remaining simple and more lightweight than other alternatives.

Real World Applications

Some of the real world applications of deep learning models are Googe Brain, Siri, Google DeepMind, Deep Face, etc. We are most familiar with those services as we use them in everyday life. Like we mentioned before there are various medical applications of deep learning models such as robotic surgeries.

In the automotive industry, deep learning methods are used in self-driving cars. In the future, it might be possible for deep learning models to integrate into automated systems of driving in order to interpret and detect sounds and sights which might be beyond the capacity of human being.

Military also uses deep learning models for building drones. In computing science surveillance deep learning models are used and play an important role. In the future, it might be possible for computers to interpret and sense with a human-like accuracy which would make a great change for surveillance systems. Optical Character recognition systems also use deep learning models that are scanning various images. Lately, it is used to read images and extract text from it which correlate to the various object which is found on images.

Other examples of deep learning models application in the real world are anomaly detection in logs and databases, the discovery of fraud in accounting, taxes, and banking. The most common application is in virus and hacking detection in the network, detection fo sensor of hardware dying in computers and various data context authentification.

Deep learning models are used in various vision algorithms. Laso in drone and missile environment are created by deep learning techniques. In industrial quality control of various typed deep learning models are used. Face and body recognition systems and various medical image analysis use deep learning methods. Any robotic navigation and satellite images analysis are dependable on deep learning models.

In sound understanding and sound filtering and editing these models must be used. Mayor sound processing software today use convolution neural network models which can edit and filter various sound features. Deep learning models can change drug design, bacteria interaction, medical diagnosis and many other scientific fields which are important for human development.

Recently, many organizations and companies like Facebook have become interested in deep learning models and methods for their particular applications. Another great name is Google which is also using deep learning models on their AlphaGo system. It is clear that deep learning models have great influence both in everyday life and in various scientific fields.

Conclusion

Deep learning is very useful when it comes to finding simple rules and solutions to overall world chaos. We are familiar now that there are three types of data, spatial, sequential and unstructured data and various method was used in the past to work on this data. However, the deep neural network is far ahead of their competition.

These older methods perform fine, but there are many limitations involved when they work on more complex data. It was often the case that the system is unable to start when it reaches an accuracy threshold. Since deep neural networks can perform on large datasets, these problems are solved.

Think of deep learning as machine perception. The perception is the power to interpret any sensory data. We do it all the time. We interpret things by giving them names of what we sensed. However, if we don't have the name for the certain thing, we arse still able to recognize it and its similarities and dissimilarities. You are able to say that two people are related even if you don't know their names, but you look at their similar features.

Algorithms perform similarly, based on perception. Algorithms are able to name things by the supervised learning and later to cluster things by the unsupervised learning. We are familiar with the difference of supervised and supervised learning, and the main distinction in supervised learning you have previously labeled training set.

Certain labels which you apply to data are the outcomes that you care about. Maybe you care about identifying people on images or identifying spam emails. No matter what you are carrying about and which problem you want to solve, deep learning models and knowledge of it is very valuable when it comes to the finding solutions.

Deep learning models are working with various algorithms and allow you to cluster, classify and predicts various outcomes. It does that simply by reading the structure, signals, and data automatically. If you train your deep learning algorithms they can make guesses and prediction about various data. Also, they can measure the error of their predictions against the training collection, and at the same time, they can correct the way in which they make predictions, so they become more accurate. This is optimization.

With deep learning, you can cluster, classify and predict anything that you have data about various time series, economic tables, the weather, images, sound, text, and DNA. Deep learning is all about what humans sense and that technology can digitize. The ability to find out what is happening in the world is much greater now as deep learning models are extensively used in various scientific fields as well as in everyday situations.

The society is able to behave more intelligently since its ability to interpret accurately what's happening around us. Prediction alone is a great power and the importance of deep learning for the society is more than obvious. The world would be certainly different if older alternatives to deep learning models are still used as before. The introduction of deep neural networks like convolution and the recurrent neural network made a great change for the better in many fields of industry and science.

After you were done with this book, you will be able to use gained knowledge to make great things. Keras is among the best high-level neural networks and gives you the opportunity to work on many different deep learning models. Whether you are going to use it for image recognition, speech to text recognition or maybe for video game development, Keras offers you limitless possibilities to explore the power od deep learning.

Analyzing Data with Power BI

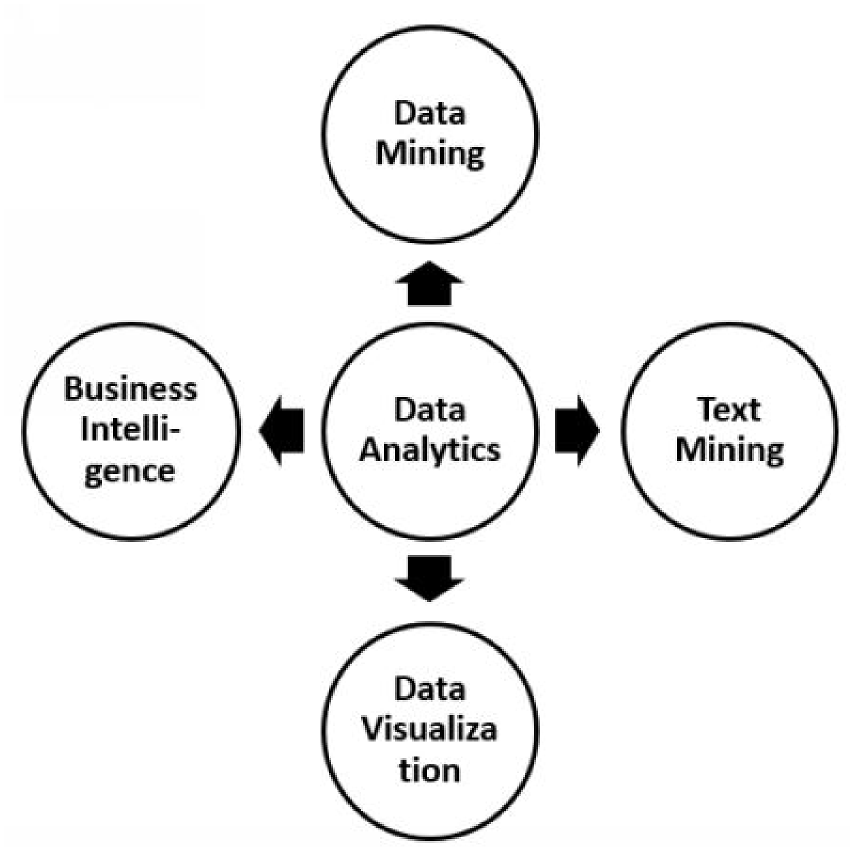
Introduction to Power BI

By Anthony S. Williams

Table of Contents

Introduction

The process of evaluating various data using logical and analytical reasoning in order to examine every single component of the provided data. This particular analysis is just one step that must be completed while conducting any research project. The first step is to gather various sources and then to analyze it forming some conclusion or finding. There are different kinds of specific data analysis methods including text analytics, data mining, data visualizations and business intelligence.



Data analysis is also known as data analytics, and the most common definition is that data analytics is in the first place process of modeling, transforming, cleansing data with one goal. The primary aim of any data analytics is to discover relevant and useful information and to come to a single conclusion and to support any decision-making. When it comes to the approaches used in data analytics, they can vary and some techniques can be used for certain data and others can’t, so the method used in data analytics will depend on the data type in the first place. Data analytics and any processes of data analytics have various facets, and a wide range of techniques is used, and these techniques have different names as well in various domains such as business science and social science.

Data Analysis Methods

Like we mentioned at the beginning, there are different methods for delivering any data analysis conclusion, and one of the is data mining. Data mining is a certain data analytics method which focuses mostly on modeling and various knowledge discovery to come to certain predictions rather than being used just for descriptive purposes. Data mining is in fact computing process used in discovering large data collections and their patterns. Data mining is eating the intersection of data systems, machine learning, and statistics. Therefore, we can conclude that data mining is, in fact, a significant subfield of computer science.

Data Mining

When it comes to the goals of data mining, the overall goal of every data mining process is discover valuable information and extract them from a data collection already provided. Then this discovered information is being transformed into an easily understandable form for further use. Besides raw analysis in the core of data mining, it also involves data management, data processing models, interestingness metrics, inference considerations and complexity considerations.

Also for data mining processing of discovered form and structures and their visualization is key towards successful data analytics. It is correct to say that data mining is a great step in data analytics as the knowledge discovery in data collections.

Data Mining		

Artificial Intelligence
Statistics
Machine Learning

So the ultimate goal of data mining is to gather profound knowledge of various provided data and to extract patterns of large collections of provided data. We also can say that data mining is buzzword which is often applied to various forms of large-scale data and information collection processing, warehousing, extraction, and statistics.

Any application of support systems in computer technology, such as artificial intelligence, business intelligence, and machine learning need data mining techniques and methods. For data mining, various terms are used as large-scale data analytics, but when it comes to the actual data mining techniques, terms like machine learning and artificial intelligence are more appropriate.

The data mining process is actually a semi-automatic analysis of a large collection of data in order to extract previously unfamiliar and unknown, distinct patterns of groups of data records or known as cluster analysis, various anomaly detection or unusual records and sequential patterns or dependencies. This usually requires certain techniques like spatial indices.

This pattern which is obtained may be seen as a type of summary of the data provided and may be used further in analyzing data. For example, it is very common in predictive analytics used in machine learning. Further, data mining process may identify numerous groups in the data provided, that can be used to gather more precise and accurate probability results by another technique decision support system.

It should be noted that neither the data preparation, results of interpretations and data collection is a part of the data mining process, but they most certainly belong to the overall process of data analytics as additional steps towards a conclusion.

Terms such as data fishing, data dredging, and data snooping are often used to describe certain data mining methods which involve large collections of data which are often too small for reliable and adequate statistical inferences which would be made in order to discover the validity of certain gathered patterns. However, these methods of data are very often used in designing new hypothesis which is used to test against the larger data collections.

When it comes to the statistical applications of data analytics, it can divide into, descriptive statistics, confirmatory data analysis, and exploratory data analytics. Exploratory data analytics is focused on discovering new characteristics in the confirmatory data analytics, or counterfeiting already created a hypothesis.

Also focused on applications of statistical models is predictive analytics often used in predictive forecasting and classification. On the other hand, text analytics is applied to linguistic, statistical and structural techniques in order to classify and extract various information from any textual source and to obtain a species of data which is unstructured.

Data analytics is closely connected to data visualization. Data integration and data dissemination. Therefore, term data analytics is often used to describe any data modeling. Text analytics term is also known as text mining is the process of gathering high-quality various information from any text source. This high-quality information gathered from the text are typically derived through the devising of various trends and patterns which correspond to different terms including statistical pattern learning.

Text Mining

Text analytics often involves the process of creating the input text and restructuring. Parsing is very often used which means that along with some derived linguistic characteristics, and in the same time with the removal of some of them, certain subsequent features are inserted into a collection of a database.

Text analytics gather these structures linguistic models and derive certain patterns which are withing the already structured data. The final step is data evaluation and interpretation of the result or output.

Text Mining		
Computational Linguistics	Inferential Statistics	Machine Learning

High-quality in text mining means that certain combinations are involved such as novelty, relevance, and interestingness. Text analytics processes often include methods such as text clustering, text categorization, document summarization, entity or concept extraction, production of various granular taxonomies, learning connection between named entities and sentiment analysis.

Text mining also involves various studies like lexical analysis, information retrieval, information extraction, annotation, pattern recognition which all help to study the frequency distributions of words in any text source.

The overall goal of any text analysis is to turn text source into structured data which can be further analyzed via different applications including natural language processing and other analytical methods. Text mining is also involved in predictive analytics and data visualization.

So text mining has a predictive purpose, and its typical application is to gather and scan a collection of documents which are written in natural language and to further model the gathered document which is set to undergo some predictive classification.

Data Visualization

Data visualization is another data analytics method often referred to as equivalent to visual communication. Data visualization involves the designing and study of various visual representations of data gathered. In other words, data visualization means that information that has been gathered is further abstracted in schematic form which includes variables and attributes for each unit of information.

The main goal of data visualization is to gather information and communicate efficiently and clearly via various plots, statistical graphs, and information graphics. In data visualization, numerical data also can be encoded using lines, bars, and dots in order to communicate visually any quantitative message. Effective and clear data visualization also helps users to analyze any evidence and data and even more complex data collections are very accessible, usable and easily understandable.

Data Visualization			
Function	Integrity	Interestingness	Form

Often users encounter on certain analytical tasks like understanding causality or they have to make certain comparisons or to create various principles of the graphic which often follows the specified task. These graphs may include showing various comparisons or showing causality between structured models. Used for these tasks are tables where a user can look up certain measurements, and charts of various kinds are used to represent relations and patterns in the data collection for single or multiple variables.

It should be noted that data visualization is at the same time both science and are, often referred to like a certain branch of statistical description. Data visualization is also grounded theory of developing various data analytic tools as well. Data created by various Internet activity is rapidly increasing, and an expanding number in the environment of various sensors is referred to Internet of things or big data also used by data visualization.

Communicating, processing and analyzing these large collections of data represent analytical and ethical challenges for this branch of data analytics. Often data scientist is called to help out in solving these challenges, and the overall science is referred to as data science.

Business Intelligence

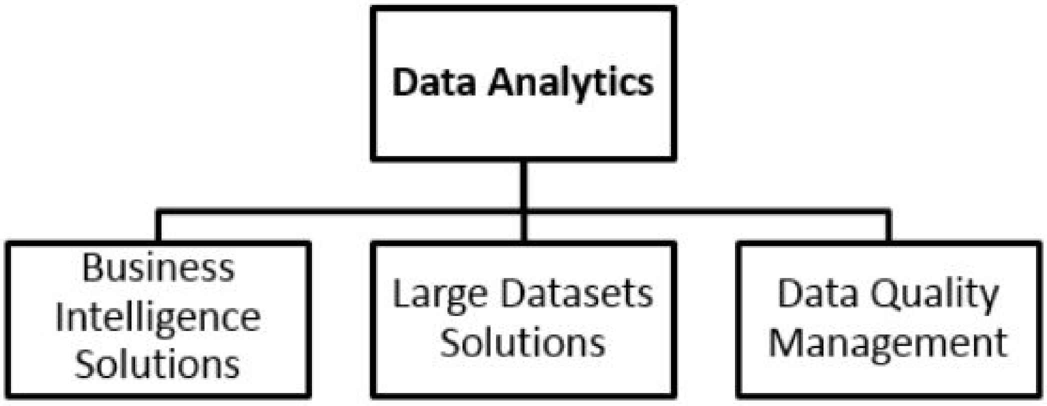
Business Intelligence is a significant part of data analytics often referred to as simply BI. Business intelligence comprises the collection of processes, strategies, structured data, various applications and technical and technologies structured models that are used by enterprises in order to support the datasets, analysis, presentation and dissemination of any business information.

Business intelligence has a great impact and significant role in current and predictive views of numerous business operations which are performed both today as well as in the past.

The most common goals and functions of business intelligence include online analytical processing, reporting, process mining and data mining, benchmarking, business performance managing, prescriptive analysis, text mining, complex event processing and predictive analysis.

Methods and technologies which are involved in business intelligence can handle a large collection of both structured and unstructured data in order to help identify, create and develop new strategies and new business opportunities.

The main goal of business intelligence involved in data analytics is to create an easy interpretation of large data collections. Business intelligence can provide businesses with very competitive market advantage by identifying better opportunities and by implementing effective techniques and strategies which are based on previous insights. Effective strategies within a very competitive market also provide long-term stability.



Business intelligence as a very efficient method of business related data analytics is commonly used by enterprises in order to support plenty of decisions related to business which ranges from strategic to operational. Most common operating decision include product pricing and product positioning on the market.

Strategic decisions related to business involve various priorities, aims, and directions regarding the broadest level. In most cases, business intelligence is most effective when it comes to the combining data that is derived from the current market in which a firm operates. These data used by a company is external data which is operated within internal sources of a company to the various business related sourced like internal data or operations data and financial data.

To get more proper and clear overall picture inner and external data are combined, and by combining these two sources, intelligence is created which cannot be gathered by any singular collection of data.

Business intelligence is also widely used a tool by empowering organizations to gain a better insight into current and new markets, to assess suitability and demand of services and products for different market fields and to measure the impact of various market efforts.

Business Intelligence applications are often used to provide data which is gathered from a large data warehouse of from a data mart, and these two concepts are commonly combined as BIDW. A large data warehouse is used to facilitate decision support since they contain copies of analytical data.

Business Intelligence	**Data Sources**	**Business Solutions**
Better understanding of business	Reducing risks of bottlenecks	Identifying waste in the system
Improving decision making process	Easy sharing and accessing information	Real-time data analysis

Chapter 1 Data Analytics Process

Data analysis is breaking a whole into its components which will be further examined separately. Data analytics is, in fact, a process for obtaining necessary data components and transforming it into various information which is useful and significant for any decision-making. In other words, any data is analyzed with the main aim which is to answer important questions, to test a certain hypothesis or to disprove theories.

Data analytics is defined as a process for analyzing large datasets, methods for interpreting obtained results of such procedures, particular ways of planning the gathering of data to make its analytics more precise, easier and more accurate. All the results, solutions and machinery of mathematical statistics are applied to analyzing data. The process of data analysis requires several different phases and often a feedback form later phases result in some additional work in phases earlier performed.

Data analysis processes require certain steps towards outcome or solution. The first step includes data requirements since the data which will be input to the analytics is specified upon certain requirements of those which are directing the analytics or customers who will use the product of that precise analysis. The common type of entity upon that the data will be gathered is described as an experimental unit. For example, an experimental unit may be a person or certain population of people who will use a product of analytics. These certain variables which are regarding a population such as variables of income and age may be obtained and specified. In this case, data can be both categorical and numerical.

Data requirements:

- *Step 1: Obtain existing relevant information*
- *Step 2: Consider obtained information needs*
- *Step 3: Specify information gaps*

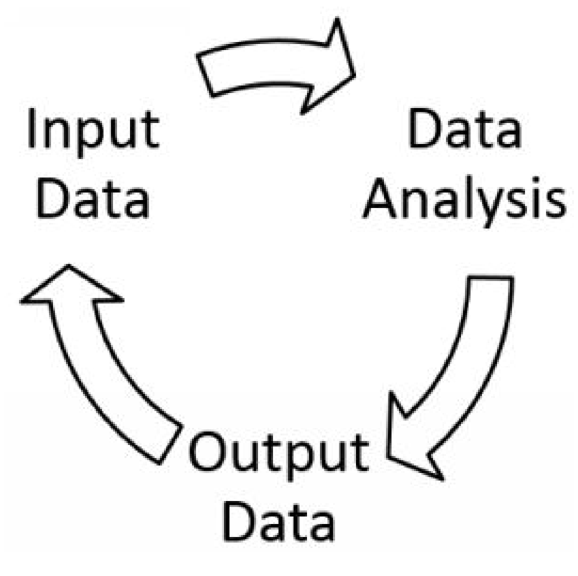
- *Step 4: Create new data and propose testing strategies*

Data is obtained from various sources. The data requirements can be communicated by analysis to keepers of the data, like information technology staff within any organization. The data also can be obtained from sensors which are placed in the environment including satellites, traffic cameras, various recording devices. Etc. Other methods of data collecting include reading documentation, online sources, and interviews.

Methods of Data Collecting:

- *Direct or interview method*
- *Indirect or questionnaire method*
- *Registration method*

Data gathered must be organized or processed for further analysis. For example, data processing may involve placing data into columns and rows if table format is used and the example of this data processing obtains structured data which is further analyzed using statistical software or a spreadsheet.



Once the data is organized and processed, it may happen that the data is incomplete or contains errors and duplicates. The need for data cleaning is significant since these problems are often, and data cleaning can solve these problems and enable data to be entered and stored properly. I other words, the process of data cleaning is methods used for correcting and preventing errors which may happen during data processing. Common tasks of data cleaning include identifying inaccuracy of data, record matching, the overall quality of obtained data, column segmentation and deduplication.

These data problems like incomplete data and various data errors also can be identified by various financial information when the totals of certain variables can be compared to independently published numbers which are believed to be reliable. The amount which may be unusual like below or above predetermined thresholds can be reviewed as well.

There are different kinds of data cleaning depending on the kind of the data which is obtained such as email addresses, phone numbers, customers, employers, etc. Quantitative methods for data cleaning for outlier detection may be used to diminish likely incorrectly obtained data. For example, textual data spelling checkers may be used to lessen mistyped words amount, but if the words are correct is harder to tell.

Data cleaning steps:

- *Import data*
- *Merge datasets*
- *Rebuild missing data*
- *Standardise*
- *Normalise*
- *De-duplicate*
- *Verify and enrich*
- *Export data*

Once the data cleaning process is done, data is ready to be analyzed. Data analysts can apply different methods and techniques which are referred to as exploratory data analytics for a better understanding of the messages and information contained in obtained data. This process of exploratory data analytics can result in some additional requests or additional data cleaning, so activities like these can be iterative in nature. Descriptive statistics like as the median or average can be generated to help better understanding of data. Also, data visualization may be used in order to examine collections of data in graphical form, and in order to obtain better insight regarding the information contained within the data.

Objectives of exploratory data analytics:

- *Discover patterns*
- *Spot anomalies*
- *Frame hypothesis*
- *Check Assumptions*

Mathematical models and formulas which are called algorithms are often applied to the data in order to identify connections between the variables like causation and correlation. Speaking generally, a mathematical model can be developed to evaluate an individual variable contained in data collection that is based on another variable within the data, containing residual errors that are depending on actual model accuracy. In other words, data is equal to a mathematical model containing an error.

Inferential statistics often include certain methods and techniques which are used in order to measure connections between certain variables in data analysis. Often used as a part of inferential statistics is regression analytics, and the regression model is used to change an independent variable in advertising or to explain certain variations in sales which are referred to as dependent variable.

If we are speaking in mathematical terms, then the dependent variable will be sales and the dependent variable will represent advertising. It may also be described as a model which is created in such way that model error minimize when the model makes a prediction for a certain range given of values of advertising. Analysts also can build models which are descriptive in order to simplify analytics and communicate outcomes.

Data modeling:

- *Explore data*
- *Condition data*
- *Select variables*
- *Balance data*
- *Build data models*
- *Validate*
- *Deploy*
- *Maintain*
- *Define success*

The next step in data processing towards the data process is to use certain computer application which obtains data inputs and produces outputs, feeding them again into the environment. Data product may be based both on algorithm and model. For example, an application which analyses data about users purchasing history of various products and recommends other products which customer might like.

Once the step of data analysis if over, the data analyzed can be reported in various formats to the users of that particular analysis which would support their particular requirements. The users also may have feedback that results in some additional analysis. Therefore, much of the data analytics cycle can be iterative.

The analyst also may consider using data visualization in order to determine how to communicate the outputs. In this case, data visualization can help to efficiently and clearly communicate the information to the users. Data visualization uses charts and tables which are proper information displays and help communicate important information within the data. Tables are really helpful to a user who is looking for certain numbers, and charts can help to explain various quantitative information which is contained in the data.

Stages of Data Processing			
Input Stage	Processing Stage	Output Stage	Storage Stage
Data collection	Performing instructions	Decoding	Storing data
Data capture			
Encoding	Transform raw data into information	Presenting data to users	Retrieve data
Data transmission			
Data communications			

Quantitative Data Analytics

There are eight kinds of quantitative data analytics which are used for better understanding and communicating from a collection of data in which particular graphs are used to help communicate the information. Users specifying requirements and analysts who are performing the data analytics consider certain information during the overall data analysis process. These eight types of quantitative messages include following:

- Time-series: One variable is captured over a period like unemployment rate over a 20-year period. Often used is a line chart which would demonstrate the trend.

- Ranking: Subdivisions are mostly ranked in descending or ascending arrangement like the ranking of certain sale performance which is, in this case, the measure. A sales person is referred to as category, and each of sales person is within categorical subdivision during a certain period. Often to show ranking, a bar chart is used since it can show the comparison between the sales persons.

- Part-to-whole: Categorical subdivisions are often measured as a certain ratio to the whole like a percentage out of hundred percent. In these situations, the best methods to show a part-to-whole is a bar chart or a pie chart since they can correctly show the various comparison of different ratios. For example, you can show the market share which is represented by competitors in the overall market.

- Deviation: Categorical subdivisions also can be compared against a certain reference like a comparison of budget expenses for different departments of a business versus actual expenses over a certain period od time. In this case, a chart is often used to show a comparison of the reference amount versus the actual amount.

- Frequency distribution: This method is used to show the number of particular variable observations for a certain interval, as the number of months in which the stock market is between certain intervals like zero to ten percent in return. A type of chart known as histogram can be used for these kinds of data analytics.

- Correlation: Also possible is a comparison between different observations which are represented by two variables in order to determine if these variable tend to move in the opposite or same directions. An example may be plotting inflation and unemployment for a certain period. To show data correlation, a scatter plot mostly used.

- Nominal comparison: When it comes to the comparing categorical subdivisions, and there is no particular order involved, a nominal comparison is used. An example may be finding the nominal comparison of the sales volume by-products codes. To show nominal comparison, a bar chart is used most often.

- Geospatial or geographic: Also possible is a comparison of different variables across a layout or map, like the unemployment rate by the number of persons or by a state. For showing messages of geographic or geospatial analysis, a cartogram is typically used.

Methods Used for Analyzing Quantitative Data

Methods used for analyzing large collections of quantitative data include checking raw data in order to find various anomalies before processing data analytics and performing significant calculations multiple times like verifying data columns that are obtained in formula form. Further methods include confirming totals which are the sum of individual totals.

Another excellent method used in analyzing quantitative data is to check the relationship between certain numbers which should be linked in a certain predictable way like ratios over a certain period. A further step includes normalizing numbers to make comparisons easier like analyzing certain amounts per relative or person to GDP. Another example may be explaining an index value which is from a base year. Next step will be breaking problems into different components by analyt+zing certain factors which led to the outcome like DuPoint analytics or certain return on equity.

Stages of Quantitative Analytics		
Framing the Problem	**Solving the Problem**	**Communicating and Acting on result**
Problem recognition	Modeling	Result presentation and action
Review of previous findings	Data collection	
	Data analytics	

To place certain variable under examination, descriptive statistics is mostly used like median, average and standard deviation. For analyzing the distribution of the significant variables, descriptive statistics are also used primarily to see how the variables cluster between and around each other. The technique that is used for dividing and breaking down of variables into smaller parts is referred to as MECE principle and after McKinsey and Company.

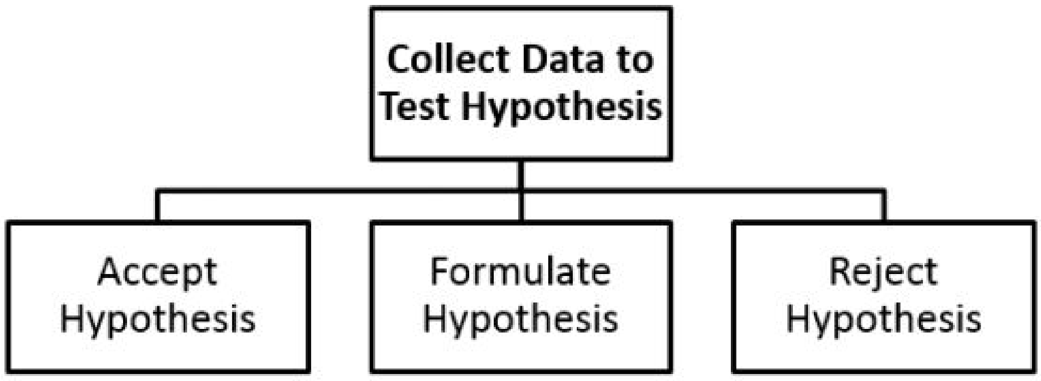
In data analysis, each layer can be divided and broken down into various independent components, and each of these broken parts has to be mutually exclusive between each other, and they further collectively place up to other layers which are above them. This relationship is referred to as MECE or in other words mutually exclusive and collectively exhaustive relationship between variables.

An example of mutually exclusive relationship may be any profit which can be broken down into independent components, or in this case, it can be broken down into total cost and total revenue. Further, total revenue will be analyzed by components like revenues of various divisions, and each of these divisions is mutually exclusive to each other. Further revenue divisions will be added to the total revenue which is collectively exhaustive in this case.

Mutually Exclusive and Collectively Exhaustive:

- *Mutually exclusive means there is zero overlaps*
  Each element is different than others
- *Collectively exhaustive means there is zero gaps*
  All possibilities are covered

Analysts also can use various statistical measurements in order to solve particular data analysis problems. Often hypothesis testing is used in cases when certain hypothesis about the actual state of various affairs is made, and the information is obtained in order to determine that actual state false or true. An example may be a hypothesis that unemployment is not affecting inflation. Therefore, hypothesis testing will consider the probability of two types of errors that are related to the data, and it's rejecting or accepting this hypothesis.



Regression analytics is also used commonly to determine the independent variable, and the extent of it affecting other dependent variables. For example, regression analytics can show us to what extent changes in the various rates like unemployment rate can affect the inflation rate. In this case, a unemployeemnet rate is an independent variable and inflation rate id dependent variable. Regression analysis creates a certain model, curve to the data or an equation line.

To determine messages and information contained withing quantitative data collections, condition analysis may ve necessary. In this case, an analyst will use condition analysis in order to determine the independent variable and to which extend it allows dependent variable. For example, condition analysis can determine to which extent is a particular independent variable such as unemployment rate necessary for a particular dependent variable or a particular inflation rate. Multiple regression analytics uses some additive logic, and each independent variable may produce the result, and the independent variables also may compensate for each other. In other words, variables are sufficient, but not necessary at the same time. Therefore, necessary condition analytics uses logic, and single or multiple variables allow the result or outcome to exist, but at the same time, it doesn't need to produce the outcome itself. Each necessary condition has to be present, but the compensation is not expected.

Data analytics techniques:

- *Regression: Linear and non-linear*
- *Classification: Supervised and unsupervised*

Data Analysis Tasks

Data analysis tasks can be organized in three main tasks which include finding data points, retrieving values and arranging data points. However, these three tasks are just main tasks, and data analysis requires other additional steps towards an outcome of the data analysis.

The first task is to retrieve values when there is already given a collection of certain cases, and attributes within these cases have to be retrieved. Further, when concrete conditions are given, using filter and analysts will find data which is satisfying certain conditions provided with filtering.

When there is already given a collection of certain data cases by computing derived value an aggregate representation in numerical form cam be obtained. Further, an analysis will find certain data cases which are possessing an extreme value of any attribute when it is ranging withing the data collection. Next step is to sort and rank a collection of data sets by some metric determined by a user.

The further range will be determined withing a collection of data cases and also an attribute that is of interest will be determined as well. The span of various values withing the data collection will be determined.

The next step is to characterize distribution withing a collection of data cases also including an attribute of interest in quantitative form. By characterizing distribution values of attributes which are over the data collection will be determined. Further anomalies which may be contained will be derived from a given collection of data cases by a certain expectation or relationship.

The next step includes clustering of data collection cases and finding similar values of attributes that are of interest. Further correlation between a collection of data cases will be determined, and attributes will obtain significant relationships among the absolute values of these attributes. The final step is to find contextual relevancy in a given collection of data cases which are important to the users.

Data Analysis	
Experimental Information	Measurement Physics
Data and Analysis Tasks	Analyst Methods and Algorithms

Experimental information: Contains details of the experiment

- *Measurement physics: Contains information about the underlying physics such as sources of noise or type of signal*

- *Data: Actual data to be analyzed including metadata*
- *Analysis tasks: Certain problem to be solved*
- *Analysis techniques: Algorithms used to solve the problems*

Data analysis methods:

- *Summarizing the data: Graphs, summary tables, descriptive statistics*
- *Finding hidden relationships: Grouping, searching, correlation statistics*
- *Making predictions: Mathematical models, inferential statistics*

Chapter 2 Fundamentals of Data Modeling

Data modeling is actually software engineering involved in creating a certain data by applying formal techniques to be further used in an information system. A data model has abstractly created a model which organizes certain components of data, organizes and standardizes them in order to see what is a relationship between them and how they relate to each other. For example, a data model can specify which data component can be composed of a numerous number of other different components.

This term data model is often used in two distinct but related senses. Often data model refers to a formalization of elements in abstract form and relationships which were obtained in a certain application field. For instance, the users or customers, products, and arrangements which are found in a manufacturing company. On the other hand, this term also may refer to a collection of concepts that are used in defining various formalizations. For instance, a data model may be created involved in various concepts like attributes, entities, tables, and relations. Therefore, the data model of some banking application can be defined by using this entity-relationship model.

A data model determines the components and overall structure of datasets. They are specified in certain data modeling notation that is most commonly in graphical form. A data model is also often referred to as data structure, specifically when it comes to the context of computer programming languages. On the other hand in the context of models used for enterprise, a data model is often complemented by additional function models.

The main function of an information system is to manage a large amount of both structured and unstructured data. In fact, data models describe the overall structure, various integrity aspects which are stored withing data collections and they manipulate various components found within data models. They do not typically describe data that is not structured including email messages, word processing documents, digital video and audio, and pictures.

The Main Aim of Data Models

When it comes to the mains of data models, the main aim is to support and back up the development of various information systems by gathering and obtaining the format and definition of data. Data modeling is consistently done across various information systems, and further data compatibility may be achieved. In cases when same data components and structures are used for both storing and accessing data, multiple applications may share this data.

The results of sharing data with different applications may cost more money since more funds are required to operate, maintain and build data models. Greta cause of this may be that the quality of created data models which is implemented in information systems may be poor regarding the interface. Therefore, often some problems may occur during data modeling process. For example, business rules which specify performance of certain things in a certain place, very often are fixed withing the data model. In other words, very small changes in the way of conducting business may lead to great charges in computer interfaces and systems.

Another problem which may occur is when entity types are not identified, or they are identified incorrectly. In this case, replication of data may occur, ad functionality of data and overall data structure can cost more money since that duplication occurred. The overall development of data model and its maintenance will cost more than it should.

For different systems, data models are different, and the result of different data models is that interfaces which are complex will be required between these systems which share data. In other words, these interfaces will sometimes cost up to seventy percent of the overall cost of systems. Another problem which may occur is that the data is not provided in order to be shared electronically with suppliers and customers. In other words, the data structure of this model has not been designed in standard form. For instance, data designed for engineering and drawings for various process plant often is exchanged in paper form.

The main reasons for these problems is a lack of data modeling standards which can ensure that all data models designed will meet requirements such as business needs and at the same time stay consistent. A data model determines the overall structure of provided data, and most common applications of data models include a design of information systems, enabling database models, enabling an exchange of data. Data models are most commonly specified in the certain data modeling language.

Types of Data Models

There are three fundamental kinds of data models conceptual model, logical model, and physical data model.

Conceptual Data Model

- This data model is used for describing the semantics of a certain domain which is being a scope of that particular data model. For instance, it can be a model of an industry or organization regarding their area of interest. This model consists of various entity classes which are representing types of things which are significant in that particular domain. Also, this model describes connections between assertions which are about certain associations among pairs of that entity class. The conceptual schema is used in order to specify the types of propositions and fact which can be expressed using this model. For instance, this model can define various expressions withing artificial language which contains a scope limited by model scope.

Logical Data Model

- The logical data model is used in order to describe the semantics which is represented by a particular technology for manipulating data. This model consists of various descriptions of columns and tables, classes that are object oriented, certain XML tags, etc.

Physical Data Model

- The physical data model is used for describing the physical significance by which data is being stored. This model consists of various partitions, tablespaces CPUs, etc.

Data Models		
Conceptual Schema	Logical Model	Physical Model

The importance of these approaches is that they allow these perspective conceptual, logical and physical to be independent and not related to each other. Also, storage technology is capable of changing data without doing any effects to a conceptual or logical data model. The column or table structure also can make some changes, and there is no necessarily any affecting the logical and conceptual model. In both cases, the data model structure has to stay consistent corresponding to the other data model.

The column and table structure also can be different from a translation of the attributes and classes, but ultimately these structures can obtain the objectives within classes structure. During every phase of software development, these projects emphasize the creation of a conceptual model. This kind of design may be further detailed into another logical data model. Further in next phases, this model also can be translated into another physical data model.

Therefore, it is possible to implement this directly to a conceptual data model.

Enterprise Data Model	**Conceptual Data model**	**Logical Data Model**	**Physical Data Model**
Documents the very high-level business objects and definitions. Provides wide scope to obtain a strategic view of Enterprise data.	The business key attributes and definitions of business data objects. It also shows the relationship between business data objects and broader scope. This model is also known as area data model.	Documents the business key attributes and definitions of business data objects. This model also shows the relationship between business data objects. Often it is within the scope of a detailed project.	This model shows technical design for example columns, tables, keys, foreign keys and other constraints to be implemented in the database or XSD. This model may be generated from a logical data model.

Database Models

A database model is data model used in order to determine the structure of a database in logical form. Database models fundamentally can determine the manner in which data may be manipulated, organized and eventually stored. The most common database model is the relational model that uses a table format to determine the structure of a database.

Database logical data models include:

- *Network model*
- *Relational model*
- *Entity-relationship model*

- *Object model*
- *Relational model*
- *Document model*
- *Star schema*
- *Enhanced entity-relationship model*

Database physical model includes:

- *Flat file*
- *Inverted Index*

Besides these common database models, there are other models which can be implemented such as correlational model, multivalue mode, associative model, semantic model, triplestore, XML database, named graph, multidimensional model, and others. A obtained database system management can provide single or multiple models. In fact, the optimal structure often depends on the organization of the data applications as well as applications of data requirements that include speed or transaction rate, scalability, reliability, maintainability, and cost. A great number of database systems of management are built on one specific data model, but it is also possible to create an offer of products which can support multiple database models.

Many physical database models, in fact, can implement various already given logical database models. Major database software actually will offer you some control in managing the overall physical implementation, and the choices which are made have a great impact on overall performance. A database model is both a way of designing and structuring data as well as a way of defining a collection of operations that may be performed on the data. For example, the relational model may define various operations including project join and select. However, often database operations are not explicit in a certain query language, but they still produce the ground foundation on that a query language is performed.

Flat Database Model

The flat database model is also referred to as flat file database or spreadsheet. This model usually consists of a two-dimensional or single-dimensional array containing data components where all elements of a column are mostly assumed to be values which are similar. On the other hand, all components placed in rows are assumed to be in some relationship to each other.

For example columns of password and names which may be used as a domain of a security database. I this case each row will have a certain password which is associated with a single user. At the same time columns of the spreadsheet will often contain a type of association with them that is defining them as data in characterization from including integers, information about date and time, floating point numbers, etc. This tabular representation of database models is, in fact, a precursor of most commonly used today relational model.

An example of flat database model:

	Route No.	**Miles**	**Activity**
Record 1	1-95	12	Overlay
Record 2	1-495	05	Patching
Record 3	SR-301	33	Crash seal

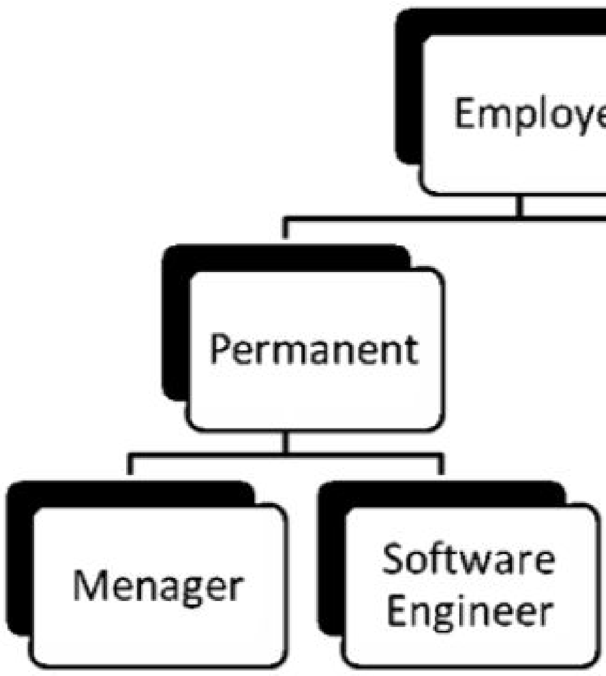
Hierarchical Database Model

Models like hierarchical database model were mostly used in the 1970s, but these models are still found today in various legacy systems. They are primarily characterized as being navigational containing strong relationships amongst their physical and logical representation while at the same time they are data independent.

In this model, data is organized in such way of a tree-like structure with an implication that there is a single parent for every record contained in the model. A sort field is, in fact, keeping related records in a certain arrangement. These models were mostly used in the early stages of mainframe management systems like the IMS or Information Management System or IBM which now uses the XML database models. This particular model allows many multiple relationships between various kinds of data obtained. Also, this model is very useful and efficient when it comes to the describing various relationship in the real world like a table of contents, any sorted and nested information, recipes, ordering verses and paragraphs and others.

This particular model is also used to describe a physical arrangement of multiple records in storage. The actual record accessing is done in a way by navigating through the structure of data by using pointers which are combined with accessing in sequential form. However, because of this hierarchical structures are no efficient when it comes to the particular database operation when there is no a full path included for every single record contained there. These limitations later have been compensating by IMS forms since additional logical models have been imposed in the ground physical hierarchy.

An example of hierarchical database model:



Network Database Model

The network database model is in fact expanded hierarchical model based on its structure. The network model allows you to represent multiple relationships also in a tree-like structure containing numerous parent components. This model was widely used before it has been replaced in certain fields by the relational model. The network model uses two fundamental principles when organizing and classifying database structure. These two fundamentals are referred to as sets and records. Records are those which contain certain fields that are allowed to be organized in hierarchical order. Network database model sets, on the other hand, defines multiple relationships amongst the record meaning there is one parent record and many sibling records. In fact, a single record can be a parent in multiple collections of sets and in the same time it can be a component of that collection as well.

A network database model set often includes linked lists in circular mode, and a single kind of record, the collection parent or owner may appear once in every cycle. On the other hand, the subordinate or child of parent record can appear numerous time in every cycle. In other words, a hierarchy can be established between two or more record kinds. For example, the type of collection can be defined at the same time of defining another collection where one owner is also an owner of the another record. Therefore, all the collections may comprise directed graph, or in other words, it can comprise ownership that is defining a certain direction. It also can comprise a certain network construction. Accessing to multiple records is both sequentially or navigationally when it comes to the linked lists in a circular format.

Network database model is also used when it comes to the representing various redundancy in data way more efficiently than in previous hierarchical database model. Also, there can be involved more than a single path from a node of an ancestor to a descendant record. The operations involved in network models are mostly in a navigational form where a certain program maintains a certain position and further navigates through the records by following their relations in which parent and child records participate. Another way of locating the records is by providing key values.

Network database models also can implement the collection of relations by using pointers which can address directly the location of any record located on disk. This gives a user a great retrieval data performance and the expense in on operations like database reorganization and database loading. Many object databases, in fact, use the navigational principle to provide faster navigation between objects and record, often using record identifiers as pointers to related records.

Relational Database Model

The relational database model is most commonly used model introduced in 1970. This model enables certain ways in data analysis which make database systems of management more independent of various applications. This model is a mathematical model that is defined in both terms of set theory and predicate logic. Systems that are implemented in relational database models are used by midrange, mainframe and microcomputer systems.The products obtained with this database model are often referred to as databases in relational terms and implementing this model is in fact approximation of another mathematical model referred to as Codd model. For implementing relational model, three terms are used, and those are domains, attributes, and relations. The relation is represented as a table containing rows and columns. These columns which are named are attributes, and the domain is the collection of certain attribute values which are allowed to be taken.

Therefore, the ground data structure of any relational model is a table containing rows and columns, and information about a certain entity such as employee will be represented in table rows which are also referred to as tuples. Therefore, in relational database model the relation, in fact, refers to the tables contained in a certain database or we can say that relationship is also a collection of tuples. The columns are those that enumerate multiple attributes of various entities like address, a name of an employee, phone number and other. On the other hand, a row will represent a specific instance of various entities such as a particular employee. Further, each tuple containing employment table will represent multiple attributes related to a certain employee.

All relations including table relations in this kind of database model have to stick to fundamental rules in order to qualify as relations. The first rule is that the arranging of columns is always immaterial in a table form. The second rule is that there are not allowed same tuples placed in rows in a table format. The third rule is that every tuple has to contain a certain value for every of related attributes.A relational database model is allowed to contain various tables, and these tables can be similar to those of flat structure models.

On the other hand, one of the various strengths which are related to this model is that any value that is appearing in multiple records and belonging to different or same tables can imply relations among these records.

However, to enforce integrity constraints, relations amongst the records also can be referred to as explicitly. Therefore, the relation by identifying parent-child record relation may be explicit as well, and values are assigned cardinality in the range one to one. Tables also are allowed to have a single attribute or a collection of an attribute which is able to act as a certain key that can be further used to uniquely identify every tuple contained in the table.

A certain key which can be used to identify a row in a database table is referred to as the main key. Keys are most commonly used in order to combine or join data contained in multiple tables. For instance, an employee database table can contain a certain column that is named as a location which would be containing a particular value that is matching the key located in a location table. Also, keys are very important when it comes to the creating of indexes that can facilitate easy retrieval of various data from large collections of tables.

It should be noted that any column is allowed to be the key, and multiple columns also can be the key. Multiple columns have to be combined into a single compound key. However, you don't have to define all possible keys in advance, since the column is allowed to be used as a single key even though originally it was not meant to be one.

There are also keys that have real-world or external meaning, and those are referred to as natural keys. If there is no any natural key that is suitable, a surrogate or arbitrary key also can be assigned. However, in major practices, most databases contain both natural and generated keys. Generated keys also can be used internally to create relations between rows which cannot be broken. On the other hand, natural keys also can be used, but less reliably, for integrations and searches with other provided databases.

Example of relational database model:

Key = 24

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack selling

⇩

Activity Code	Date	Route No.
24	01/12/01	1-95
24	02/08/01	1-66

⇧

Date	Activity Code	Route No.
01/12/01	24	1-95
01/15/01	23	1-495
02/08/01	24	1-66

Chapter 3 Getting Started with Power BI

In this chapter of the book, we will get familiar with the powerful tool provided by Microsoft Power BI. The Power BI is great business analytics tools which can deliver significant insights throughout your business organization. Power BI enables you to link together hundreds of different data sources, drive ad hoc analytics and to simplify and categorize data preparation. Power BI powerful business analytic tool which is allowing you to produce outstanding reports and then publish those reports for your business organization to have a look at them both on the Web and portable devices.

As we said, Power BI by Microsoft is outstanding business analytics tools, and business analytics refers to the practices, skills, and techniques that are used for iterative investigation and exploration of business performance in the past. These gained perspective of past performance is a key towards gaining current insight and proper driving current business planning.

Business analytics is mostly focused on developing better and new insights of business market and a better understanding of overall business performance that is based on statistical methods and data analytics. On the other hand, business intelligence is focused on a consistent collection of metrics in order to measure the performance in the past and to guide future business planning that is also based on statistical methods and data.

Business analytics extensively use various methods of statistical analytics like predictive modeling, explanatory and management based on fact to drive certain decision-making. Therefore, business analytics is also related to management science. In this field, it can be used an input for fully automated decisions or decisions made by a human. Business analytics is, in fact, reporting, querying and online analytical processing which is all provided by the Power BI tool compatible both with a desktop computer and portable devices users.

Reporting, querying, alert and online analytical processing tools like Power BI have a great impact on overall business planning and business management. These tools can answer important questions like how many happened and what happened, where the problem is, what may be the solution and which actions are needed. Business analytics tools provide these answers as well as they can predict what will occur next and what is the best which can occur. So certain prediction and optimization can be developed.

Business analytics answers the questions?

- *What happened?*
- *When did it happen?*
- *Why did that happen?*
- *Will that happen again?*
- *What will happen if we do something differently ?*
- *What is the data telling us?*

Types of business analytics include decision analytics which is used to support decisions made by humans with visual data analytics which the models use in order tor effect reasoning. Another type of analytics important to business analytics is descriptive analytics used in order to gain a better insight from past data with clustering, reporting, scorecards, and others. Predictive analytics is also an important field related to business analytics which uses predictive modeling based on machine learning and statistical techniques. Another important field of analytics is prescriptive analytics which recommends various decisions using simulation and optimization.

Business analytics includes:

- *Statistical and quantitative analysis*
- *Data mining*
- *Multivariate testing*
- *Predictive modeling*

- *Text analytics*
- *Big data analytics*

Power BI is a great tool both for analysts and business user. With Power BI you can transform and clean data simply, and data cleaning may be time-confusing using different software. Tools for data modeling and data shaping are extremely easy to use and it they will save you many hours in your busy day. Power BI is a powerful solution for data analysis, and its capabilities allow you to visualize, connect, shape and share data quickly. When it comes to the delivering data to decision-making with Power BI, you are able to share and publish interactive reports on various platforms.

Power BI lets you to easily go from any data to insight to final action. You can create reports in minutes and connect to hundreds of various sources. Power BI software is also compatible with portable devices so you can view your dashboard at any time on the web both via desktop computer and portable device. This tool offers a simplified management, and your data is completely secured as soon as you publish it.

So for you to start, you should first get to know better what this tool offers you and how it can significantly improve your business decisions. Power BI is a wide collection of software apps and services which are working together in order to make your data into visually immersive, coherent and interactive insight. Power BI allows you to easily connect to various data sources, discover and visualize them and publish and share them on the Web.

Power BI also lets you create quick insights from various Excel spreadsheets as well as from local databases. This tool is always ready for real-time data analysis and data modeling, which makes it your personal visualization and reports tool serving as a great tool for entire corporations, group and individual projects and various divisions. Power BI tool consists of three elements, the desktop version, mobile applications and the service so that you can use Power BI on the go as well.

Therefore, you can use Power BI for various data analyzing tasks including business reports, number-crunching, monitoring progress on your sales, creating engaging business reports, getting better insight into the current market and others. When you create reports, that report will be published on Power BI service and shared to other users who then can consume information.

Everything you do with Power BI may be broken into a ground building blocks. Once you get to know these building blocks, you can further expand every of them and begin creating more complex reports.

The Fundamental Building Blocks:

- *Visualizations*
- *Datasets*
- *Reports*
- *Dashboards*
- *Tiles*

A visualization is often referred to as simply visual. Visualization is, in fact, a visual data representation like a color-coded map, chart and graph and other types of data which is possible to be represented in a visual format. Visualizations may be simple in a form such as a number which represents something important, or more visually complex such as a gradient-color map which shows sentiment towards a particular social concern or issue. The ultimate goal of any data visualization is to represent data in such way that it provides various insights and context, which is difficult to obtain from a raw table containing text or numbers.

Dataset is already familiar to you, and Power BI uses its data collection in order to create visualizations. As a dataset, you can use a single table created in Excel, or you can use a dataset which is a combination of various sources which you can further combine and filter in order to create a unique dataset to be further used in Power BI.

For instance, you are able to create a dataset which is provided from multiple different database collections, like an Excel table, a single website table and online results of some marketing campaign. Therefore, that uniquely designed database is still referred to as a single dataset even though it is combined from multiple sources.

With Power BI you can also filter your data, and this tool allows you to focus on those features which are important to you. For instance, you can filter contact database in such way that only those customers who eventually received emails will be included in that specific dataset. Also, you can create visuals that will be based on that specific subset of users who were previously included in your campaign. Filtering data generally helps you to focus on things that are important to you and to emphasize your overall efforts.

This powerful business analytics tool also lets you use a multitude of various data connectors which may be included. Whether the data you want to obtain is in SQL database, Oracle, Azure or Excel as a spreadsheet these already built-in connectors of data will allow you to easily connect various sources to that specific data, and filter it if needed and eventually bring it into your created dataset. Once you gained a dataset, you can start working of visualizations which can display various portions of a different dataset in numerous ways.

A report is a collection of various visualizations which appear combined on a single or multiple pages. Using Power BI, you are able to create various reports such as those reports used in sales presentations. In other words, a report in Power BI is a certain collection of various components which are all related to each other in some way. For creating reports, you can also use Power BI service. Reports allow you to design multiple visualizations on a single or multiple pages if needed and of course, you can arrange them in any way you like. Therefore, you might want to create a report on quarterly sales, various product growth in a certain segment, reports based on migration patterns or anything you can think of. Whatever your topic may be, reports allows you to combine and organize wide range od visualizations into a single one.

Once you are ready to share a collection of your visualizations or to share pages from a report, you will create a dashboard. Pretty much like any dashboard form, Power BI dashboard is a certain collection of various visuals which can be shared with another user via Power BI service. Often a dashboard is a particular group of various visuals which provide quick and easy insight into your data which you want to share and present to others. A dashboard which has to be placed on a single page is called canvas. Dashboards also can be shared for other people to interact with them via their portable devices.

Another important fundamental element of Power BI tools us tile which is, in fact, a single visualization that is found in created reports or within a dashboard. You will see that a tile is a rectangular box which contains independent visuals. When you are using Power BI for creating both reports and dashboards, you are allowed to arrange and move tiles in whatever direction you want. Also, you can change tiles and make them bigger or adjust their width and height as well. You can also interact with various tiles which are not created by you, so this is called as consuming or viewing a report or dashboard. When you are just viewing or consuming these elements you are not allowed to change the way in which they are ordered.

Power BI Service

As we are already familiar, the common flow in Power BI is to create a report, or a dashboard which will be shared and published on Power BI service and other users will be able to interact and view your work both on a mobile app and in the service.

Power BI service includes a content pack which is a collection of previously configured visuals which are ready-made base on various data source like Salesforce. This content pack offers you a wide collection of various entrees which are created to go together. Content pack reports and dashboards are presented in a ready-to-go way all well combined for further use.

You can get data from the Power BI service with just one click, so you just have to select the Get Data located in the bottom corner. You will see which available sources you can use with some additional sources like Azure data and Excel databases and files. Power BI also lets you connect to specific software service known as SaaS cloud services or providers which are Facebook, Salesforce, Google Analytics and others. From these specific software services, you can get an already ready collection of visually previously arranged in reports and dashboards. These are called Content Packs. Content packs pre-arranged let you get started instantly and just select the service you want.

For instance, if you are going to use the Salesforce pack, Power BI will connect you to your account at Salesforce, and then pre-defined collections of various visuals presented both in reports and dashboards will be available to you for further use.

Power BI offers you various content pack sources, and you will see that these services are arranged in alphabetical order so you can easily find the service you want. Once the data from chosen pack is loaded, you will see the content pack of dashboards and reports.

In addition to provided reports and dashboards, a dataset is also provided which contains the certain collection that is pulled from service pack you chosen.

Power BI Service:

- *Software as service SaaS*
- *Hybrid Solution on-premise or cloud data sources*
- *HTML5 reports and dashboards supporting mobile apps*

- *Real-time and streaming data source support*

On the dashboard which is provided, you can click every visual, and there contained any you will be automatically taken to other page containing reports of that certain visual.

Another great thing provided in Power BI service is that you can also ask various questions of data, and you can also design visuals which are based on the question you asked in real time.

Once you notice the visual which you like you select Pin icon which is located to the right of Natural Language Bar, and that certain visual will be placed on your dashboard.

Other options which are provided in Power BI service is refreshing data contained in the content pack or any other data you are using. In order to refresh data, you select the three dots which are located next to data set, and you will see that menu appears.

Next step is to select Schedule Refresh option located at the bottom of the menu. Further, the selecting dialog will appear, and you may refresh any data you want.

Power BI Steps:

- *Bringing data into Power BI a creating a report*
- *Publishing the report to the Power BI service where you will build dashboards and design new visualizations*
- *Sharing your dashboard with other users*
- *Viewing and interacting with shared reports and dashboards using Power BI service*

Chapter 4 Analyzing and Visualizing Data with Power BI

The majority od Power BI features are both available fro Power BI desktop version and Power BI service. We are already familiar with getting data, but it may often occur that obtained data is not well-formed or is not clean as it should be. In this case, data which is gathered have to transform or cleaned which is known as data cleaning and data transforming.

Power BI both desktop version as well Power BI service allows you to visualize, clean and connect your data. Using this tool you can further visualize and model data in various ways. Once you downloaded and installed Power BI application to your desktop computer, you are ready to go.

This tool lets to connect a wide range of different data sources, from anywhere in local areas to Excel spreadsheets to cloud service. Features like formatting and cleaning data are well performed which make data usable in various fields. Renaming and splitting columns, working with various dates and changing data types also can be performed. You also may create various relations between columns to make it easier to further model and analyze data.

Once you opened Power BI software on your computer, to get started select Get Data. Once you do that the collection containing data sources will appear and you can choose from offered data sources. No matter what data source you decide to work with, Power BI will connect to that particular data source, and you will see which data is available from that chosen source.

In order to start building reports, you should select Report view. The report view contains five areas including the ribbon, the report view, the pages tab, the visualization pane and the fields pane.

- The ribbon: Displays tasks which are associated with visualizations and reports
- The report view: In report canvas visualizations are arranged and created
- The pages tab: The pages tab allows you to add or select a report page
- The visualization pane: In visualization canvas, you can customize axes and colors, drag files, apply filters and change visualizations
- The fields pane: Query filters and elements can be dragged to the Filters are within the Visualization panel or dragged onto the Report view

The fields and visualization may be collapsed if you select the small arrow located along the edge which would constantly provide more space for building visualizations in the report view. Also, you can expand that section in addition to collapsing it. In order to design a visualization you just have to drag a field onto the report view. Power BI automatically creates a visualization based on the map, since it recognizes state field which is containing geolocation data.

Once you created your data visualization, it is ready to be published. To do this go to the home ribbon and select button publish. You will have to sign in to Power BI first, once you do that your data visualization will be published successfully. When you sign in to your Power BI account, there you will see that you published data visualization in the Power BI service. You can go to your reports and select pin icon to add that visual into your dashboard.

Connecting Data Sources

In Power BI you can connect a wide range of various data sources. Currently over fifty-nine various cloud services like Marketo and GiHub contain certain data connectors so that you can connect to various sources through ODBC, text, CSV, and XML. Power BI tool can even scrape data directly from any website. In the same manner, first you select button get data located on the home table. There are various sources provided and in order to get started select one to establish a connection. You may be asked to find that source on a network or your computer, depending on which source you selected.

Once you connected data source, the navigator window will appear. The navigator window displays the entities or tables of your chosen data source, and by clicking on it, you can get a preview of components within data source you selected. Selected entities and tables can be imported immediately. You can also edit data before importing it by selecting Edit.

Once you have selected entities and tables, you would like to bring into Power BI, and you select load located in the bottom corner od navigator. Sometimes you may wish to make some changes to entities and tables before loading them, so to do this just select filter transform or edit button before loading data sources to Power BI.

Power BI also includes query editor which is a powerful tool for transforming and shaping data, so your visualization and data models can be outstanding. In order to launch query editor, you should select edit located in the navigator. Another way of launching query editor is from Power BI Desktop by selecting edit queries located on the home ribbon.

Once you selected query editor with data already provided and ready to be transformed you will notice various sections including sections like sharing and viewing, selected query display, a listing of query's properties, steps which are applied and others. In order to see what are available transformations, you just right-click on a certain column display. You will see that you can make changes like duplicating the column, reforming the columns, replacing values, renaming the column, etc. Also, you can split text columns into various multiples by delimiters which are common.

The query editor also contains some additional tools like changing the data type, extracting elements from various dates, adding scientific notation and others. Once you applied a transformation, each step you did will appear in the section applied steps located in the query settings canvas. This list also can be used to review and undo certain changes which you have made previously. In order to save changes select close and apply button located on the home tab. Once you do this, query editor will apply all changes which you made, and they will be applied to Power BI desktop.

Data Transformation and Data Cleaning

Once you have shaped your data using query editor, you can also look at it in different ways as well. Three views are offered in Power BI, and those are data view, report view, and relationship view. In order to access any of these views, you select icon located on the upper side of the canvas.

Power BI views:

- *Data view*
- *Relationship view*
- *Report view*

In order to change the view, you just select one of two icons remaining. During the data modeling process, Power BI can combine data from various sources into one report. You can also add a source to already create a report by selecting edit queries located in the game ribbon and further select new source. You also can import data from various files at the same time, and these will appear as binary content in the query editor.

One of the most useful tools here is filters, and for instance, you can access various filters by an opening checklist of various text filters which can be used to remove certain values within your data model. You can also combine append queries and turn multiple tables into a single table which will contain only the data you want. Using append query tool you can also add the data to an already existing query from a new table. In Power BI you can also add a custom column based on M query language terms.

Power BI allows you to import data from various data sources, its modeling and visualization tools work best with data in columnar format. It may happen that your data is note imported in a simple columnar format which is a case with Excel spreadsheets which often are not optimized for further automated queries. However, Power BI provides a tool to easy and quickly transform any multi-column data into datasets that are proper for further use.

Using transpose in query editor, you can turn rows into columns and columns into rows in order to break data into formats which can be easily manipulated. Sometimes you may also need to format data so you can identify and categorize that certain data once it is loaded. You can use fill to turn zero values into the certain values located below or above in a provided column.

Unpivot columns also can bu used to cleanse data into a dataset which you can further use. Using Power BI you ar able to experiment with numerous transformations on data, and you can also determine which types of data in columnar format can be worked on in Power BI. In should be noted that all actions you will take are going to record in applied steps sections, so you can always undo any changes you made.

Modeling Your Data

In order to create a logical relationship between various data sources, you need to create a certain relationship between these data sources. Power BI allows you to know exactly how these tables are related to each other so you can create reports and visuals. On of the many great advantages which those tools offer is that there is no need for flattening your data, and you can use various tables from different sources and further define the relationship between them. You can also make calculations and assign metrics to them in order to view certain segments of your data, and further use these metrics in visualization for much easier modeling.

Therefore, you can visually set the relation between elements and tables. In order to see a diagrammatic view of data gathered use the relationship view located on the left next to the report view. When you access the relationship view, you will see a block which represents every table and table contained columns. Lines which appear between them represents a relationship. Removing and adding relationship is easy, and by right-clicking it and selecting delete, you can remove it. Drag and drop are used when you want to show relationships between tables. You also hide an individual column or a table by selecting hide in report view.

In order to get more detailed view of data relationships, you select manage relationships located in the home tab. Once you do this, manage relationship dialog will appear where all relationships are displayed as a visual diagram or a list. You can also edit relationships manually from manage relationships dialog. You can here find cross-filter direction and cardinality direction to set your data relationships.

Options available for cardinality are one to one and many to one. In fact, many to one are dimensional type relationship, while one to one is used for connecting single entries in any reference tables.

On the other hand, cross-filtering is in one direction, and some limitations are involved regarding data modeling capabilities. It is important to set accurate relationships since they will allow you to design more complex calculations of multiple data components.

Using Power BI tool you can also create calculated columns which is an easy way of enhancing and enriching your data. A calculated column is in fact newly created column which you create by defining certain calculations which combine or transforms components of already existing data. Once you created calculated column, the next step is to optimize data models to create better visuals. Power BI allows you to optimize data which would make it instantly more usable to create visuals and reports.

Further steps include creating measures and working with time-based functions. The data analysis expression language involved in Power BI offers various useful function mostly those who are time-based like year over year to year to date calculations. Here you can define certain measure or time once and further slice it by various fields. These defined calculations are called measure. Also significant in data analysis process are calculated tables that allow you to express a wide range of new capabilities for data modeling. For instance, if you want to do various types of merge join or to simply created new tables which would be based on a functional formula, creating calculated tables is a way to go.

With Power BI it is easy to work with time-based data since the data modeling tools in Power BI can automatically include various generated fields which allow you to drill through quarters, years, months and days with only one click. Once you created a data visualization using date fields, breakdowns based on period will be automatically included. For instance, the data table can automatically be separated into a day, month, quarter and year by Power BI.

Data modeling in Power BI:

- *Managing data relationships*
- *Optimizing data models to create better visuals*
- *Creating calculated columns*
- *Creating calculated tables*
- *Creating measures and working with time-based functions*

- *Exploring time-based data*

Visualizations in Power BI:

- *Creating and formatting slicers*
- *Map visualizations*
- *Scatter charts*
- *Tables and matrixes*
- *Gauge and single number cards*
- *Waterfall and funnel charts*

Power BI			
Power Query	**Power View**	**Power Pivot**	**Power Map**
Transform, merge and filter	Dashboard dynamic power	Data model merge and format	Geo-visualization

Chapter 5 Applications of Data Analysis

Data analysis applications are used to improve and measure the performance of past and current business operations. They use collections of past data in order to provide tools and information which are useful to business users, and that will let them make significant improvements. Levels for business analytics are operational reporting, business dashboard, analytic application and analytics reporting.These applications of data analytics may further extend to a domain of predictive analysis. Business analytic applications mostly relate to analyzing business processes important in support of users decision making. For instance, a business analytic application may relate to various sales analysis, risks involved in profitability analysis or accounts analytics.

Data analysis is widely used by banks in order to differentiate among their customer based on various characteristics including credit risk. They used data analysis to match user characteristics with product offering which is appropriate for them. Major gaming firms use data analysis in customer loyalty programs. Quantitative analysis together with the predictive analysis is used by wine companies to predict appeals of wines.Basic domains of data analysis include enterprise optimization, marketing analytics, transportation analytics, telecommunication, fraud analytics, financial services analytics, pricing analytics, risk and credit analytics, health care analytics, supply chain analytics and others. Data analytics have been widely used in business management since the 19 century, but it has gained more attention in the late 1960s when a computer has been used in decision support systems. Since then, data analytics have changed and at the same time formed together with the development of resource planning. Later business analytics with the introduction of a computer has exploded, and the change has brought limitless business analytics opportunities.

Big Data Applications in Real Life

Big data is definitely taking the world, and the importance of data analytics has grown rapidly making various companies and industries rely on data analysis in order to gain insights from various data sources, improve their business performance through current data analysis. Data analytics provide great insights using both semi-structured and unstructured data sources. Also, data analytics helps companies to mitigate risks and make smarter decisions by performing proper risk analysis. Many industries are propelled by data analysis including insurance services, healthcare, public sector services, industrialized and natural resources, banking sectors, and others.

Data analysis is also important to various internet search engines like Google, Yahoo, AOL, and others. These search engines use data analysis to deliver the best possible results when we search query. Digital advertisements especially targeted advertising are using various data analytics algorithms and models in order to get higher CTR than that provided by doing traditional advertising. They use data analytics to gain insight into customer's past purchases and behavior.

Internet service like Amazon also uses data analytics to support their recommender system and make suggestions for their customer based on products they have already purchased. So they help you to find products which might interest you which adds a lot to the overall users' experience. All major companies also use this engine in order to promote their new products and make suggestions which are according to customer's interest. In order to improve overall users' experience internet, giants like Google Play, IMDB, Twitter, Linkedin, Netflix and others use this recommender system.

Various gaming software like Activision-Blizzard, Sony, EA sports, Zynga, Nintendo, and others have improved overall gaming experience using big data analytics models and algorithms which improved themselves as you move up to higher levels. Also in motion gaming, your opponent or computer can analyze your past moves and in according to them to shape its game.

Other applications of data analysis include price comparison websites, airline route planning, banking fraud and risk detection, delivery logistics, and others. From this chapter, yu can see what great impact data analytics models and algorithms have on both in various scientific fields and in real-life situations. Just imagine that Google wouldn’t be what it’s today without data analysis techniques and data analysis modeling. Other internet searches as well wouldn’t be what they are today if there is no knowledge of predictive and data analysis.

Conclusion

This book will help you to gain a better insight into the world of data analytics and with some practice, you will become Power BI expert. Power BI is very powerful tool provided by Microsoft, and you can use it for your various projects which are in the domain of business and data analytics. In the previous chapters, fundamental steps towards great business analytics solution are explained, and you will be able to create by yourself outstanding data models using Power BI. You are also able to share your models with another user by Power BI service, and you can as well interact and view solutions published by other Power BI users.

Reading this book you will also gain valuable knowledge in data analysis, and in the previous chapter, we have seen what are important applications of data analysis today major companies and industries. We have seen what is the impact of data analysis today in various services like healthcare, banking systems, Internet search engines, gaming development, business management systems, digital advertising and recommender systems, so you have a better idea of how data analysis is significantly affecting various scientific fields as well as real-life scenarios.

Here you learned fundamentals of data modeling process and how to adequately communicate with your visuals and data. Once you get to this part of the book, you will be ready to go on your adventures and ready for application of everything you have learned here to your projects. This book is a guide for you into the world of data analysis, and you will make most of it, by applying everything you have learned here and created something that is important to you and an adventure is waiting for you around the corner.

Reinforcement Learning with Python

A short Overview of Reinforcement Learning with Python

By Anthony S. Williams

Table of Contents

Introduction

Reinforcement learning is the fundamental problem when it comes to the getting an agent to act out there in the world in a certain manner of maximizing its rewards. For instance, consider a teaching job when you are teaching your dog a certain new trick. In this case, you are not sure and you cannot tell your dog what to do, but there is an option to reward it if it does the wrong or right thing. The dog, in this case, has to figure out what it is supposed to do in order to make it to the reward. The dog also has to learn what made him deserve that punishment as well. This is known as the credit assignment issue.

Remember this example with the dog and teaching it to perform new tricks since we can use the very similar method in order to train computers to perform different tasks, like playing chess or backgammon, scheduling our jobs and meetings as well as controlling robot limbs and much more. We are able to formalize the reinforcement problem as the certain environment specifically modeled as a certain stochastic finite state device or machine with particular inputs. These inputs are, in fact, certain actions sent from the specific agent. Machines also come with certain outputs as well notably rewards and observations specifically sent to the agent. We can represent it as follows:

- State transition function is represented as P (X (t) |X (t-1), A (t))
- Observation (output) function P is represented as (Y (t) | X (t), A (t))
- Reward function is represented as E (R (t) | X (t), A (t)

You can easily notice that what certain agent sees greatly depends on what it performs. In other words, it greatly reflects the fact that certain perception is an entirely active process. It should be noted that the agent is also modeled as stochastic FSM notably containing inputs including rewards and observations specifically sent from the environment. The agent also comes with certain outputs like certain action sent to the specific environment. When it comes to the ultimate goal of the agent, it is to find that policy or state-update function that will be leading it to the maximizing the expected sum of different discounted rewards. We can represent it as follows:

E [R _ 0 + g R _ 1 + g ^ 2 R _ 2 + ...] is equall to E sum _ {t=0} ^ infty gamma ^ t R _ t

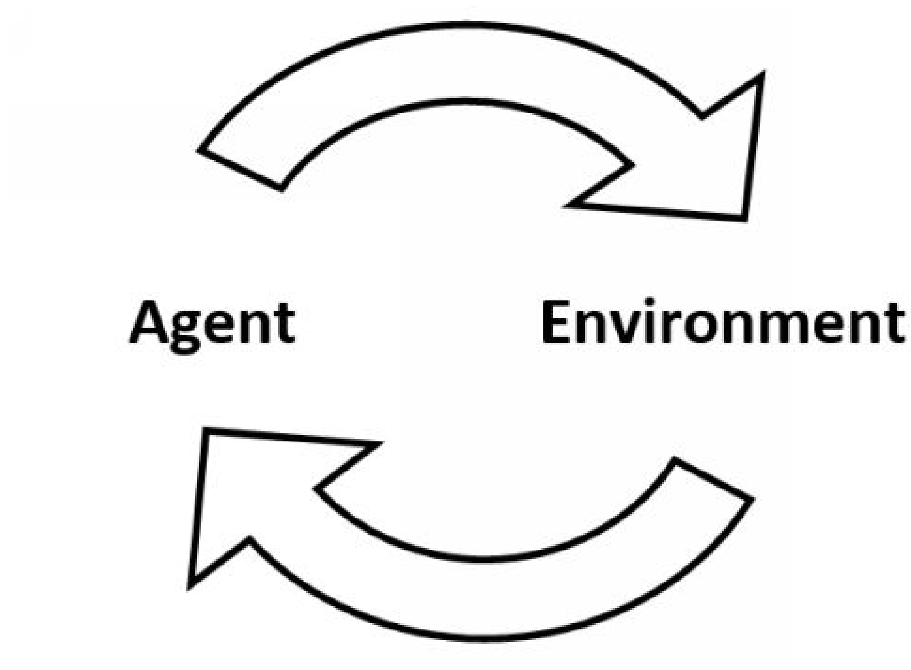
In this function, gamma is a certain discount factor that models that fact future reward notably less than certain immediate reward. Mathematically speaking, we need gamma to be less than one in order to make that infinite sum coverage. On the other hand, the environment will have zero rewards when it comes to its absorbing states.

Reinforcement is an area of a wide field of machine learning notably inspired by behaviorist psychology specifically concerned with how certain software agents might take different actions in a wide diversity of environments in order to maximize that notion particularly leading towards the cumulative reward. The main problem is studied in different other disciplines, including operations research, control theory, game theory, simulation-based optimization, information theory, swarm intelligence, genetic algorithms, and statistics. This greatly speaks about a wide range of reinforcement learning applications when it comes to the different scientific fields.

In the control literature and operations research, approximate dynamic programming are those fields where different reinforcement learning techniques are incorporated and carefully studied. Reinforcement learning problems also have been studied in other scientific fields like a theory of optimal controls. It should be noted that most studies are commonly concerned with the various optimal solutions and their existence in addition to their characterization rather than being concerned with the learning and different approximation aspects. In game theory and economics, reinforcement learning is commonly used in order to explain how equilibrium can arise under certain bounded rationality.

When it comes to the machine learning, the environment is commonly formulated as a certain MDP also known as Markov decision process as different reinforcement learning practices and algorithms for this context commonly utilized various dynamic programming methods and techniques. When it comes to the main difference between reinforcement learning and other classical algorithms, I have to mention the one regarding the latter that do not need any knowledge on MDP and they still target huge MDPs where some more exact methods simply become invisible.

Reinforcement learning greatly differs from other standard supervised learning techniques since reinforcement algorithms correct output and input pairs, but never present them. It should be noted that sum-optima actions are never explicitly corrected and presented as well. Reinforcement learning algorithms instead focus mainly on on-line performance that involves finding that balance between exploitation and exploration. In these terms, exploitation is that current knowledge while exploration is a certain uncharted territory. There is a common reinforcement method or exploration vs. exploitation relationship notably studies through common multi-armed bandit problem as well as in different finite MDPs.



Reinforcement learning is commonly used practice when it comes to the machine learning. It allows machines and software agents to automatically determine the specific ideal behavior within a particular context in order to maximize its performance. Reinforcement learning is greatly concerned with the common problem of finding the most suitable actions to take in a specific situation to maximize the most suitable reward. It should be noted that reinforcement learning algorithms are not given any specific goal. These algorithms instead are mainly forced to learn their optimal goals by error and trial.

You can think of the classic game Mario Bros. In this game, different reinforcement algorithms would learn by error and trial and determine that specific movements, as well as the button, push notably advancing the player's standing and players' score in the game. In this case, error and trial aim in order to result in a certain optimal state when it comes to the gameplay. The reinforcement learning algorithm also prophesies that interaction existing between the learning agent and environment. The environment commonly rewards the agent for any corrective action particularly the reinforcement signal. Leveraging the certain rewards obtained, further, the agent improves its particular environments knowledge in order to select the following action.

Reinforcement learning when it comes to the artificial intelligence is a certain kind of dynamic programming which, trains different algorithms mainly using a system of punishment and reward. A reinforcement learning agent or algorithm learns everything relevant by interacting with its certain environment. The reinforcement learning agent receives a particular reward by performing in a current manner and it receives penalties when performing in an incorrect manner. It should be noted that the algorithm or agent is able to learn from its environment without any intervention from human by minimizing its penalty and maximizing its reward.

Reinforcement learning is a commonly used approach to machine learning which is absolutely inspired by behaviorist psychology. You can think of the approach in a similar manner to how a child embarks on an adventure and new state in life when trying to walk or when trying to perform any new task. It should be noted that this approach contrasts other learning approaches in that reinforcement algorithm is not told hor to perform a certain task. The algorithm works through the issues completely on its own.

Chapter 1 Types of Machine Learning Algorithms

In this section of the book, you will get to know the differences between reinforcement learning and other types of machine learning algorithms. It is more than apparent that we are living in this most defining period when it comes to the human history. This the period when computing rapidly moved from certain large mainframes to desktop computers than finally to the cloud. However, what makes ti certain period the most defining, in fact, is something that has not happened yet, but it is clearly coming our way in these several years to come.

What makes this certain period exciting for someone like us, is the great democratization of the techniques and tools that followed this great boost in computing. Today, data scientists are able to build an amazing data-crunching machine with some complex algorithms in just a few hours, and that is what makes this period the most defining in the human history. However, reaching to this point in data science was not easy, and like many other periods, this period as well had some very dark nights and days.

In this section of the book, you will understand common types of machine learning algorithms compared to the subject of the book, reinforcement learning. There are three types of machine learning algorithms including supervised learning, unsupervised learning and reinforcement learning. In order to better understand the main concept behind the reinforcement learning, you have to get to know other machine learning types briefly, before we embark on serious machine learning in Python.

Machine learning comes in different flavors that depend on the algorithm or model and its overall objectives. The types of machine learning algorithms are divided into these three groups based on their purpose. It is very useful to tour different machine learning algorithms so you can devote to reinforcement learning with a basic understanding of supervised and unsupervised learning techniques.

These learning techniques are commonly grouped by the learning style and by similarity. Some of the most common learning techniques grouped by their similarity are regression algorithms, regularization algorithms, instance-based algorithms, Bayesian algorithms, Decision tree algorithms and Artificial neural network algorithms. In the following section, you will be introduced to these most common machine learning algorithms and their purpose. Think of them as representative notably very useful since you will get an idea of machine learning fundamentals that will come useful when it comes to the reinforcement learning algorithms.

Types of Machine Learning Algorithms		
Supervised Learning	**Unsupervised Learning**	**Reinforcement Learning**
Makes machines learn explicitly	Machine understands the data and identifies structures	An approach to artificial intelligence
Data with clearly defined data is given	Evaluation is qualitative or indirect	Reward based learning
Direct feedback is given	Does not predict or find anything specific	Machine learns how to act in a certain environment
Predicts outcomes		Maximizing rewards and minimizing penalties
Resolves classification and regression problems		

Supervised Learning

In this section, I am focused on explaining to you machine learning algorithms notably grouped by their learning style. There are many different ways when it comes to the training machine learning model. The algorithm also can model different problems mainly based on its interaction with the environment or experience. We will first have a look at supervised machine learning. In this certain approach, input data is commonly called as training data and it has an already known label and result like not-spam or spam or certain stock price at a certain time. A supervised learning model is prepared through the careful training process in which it is mandatory to make a certain prediction. The model is further corrected by this prediction that turned out to be wrong. The training process then continues until the algorithm achieves that desired level of accuracy when it comes to the training data. Most common supervised algorithms are regression and classification including Back Propagation Neural Network and Logistic Regression.

When it comes to the supervised learning approach, we have a certain model consisting of outcome variables and target variables. Outcomes variable is also known as a dependent variable specifically predicted from a certain given set of different predictors also known as independent variables. Using these variables, we are able to generate a function notably able of mapping inputs to every desired output. The training process than continues.

Supervised learning happens when a model learns from certain exampled data and when a model is associated with certain target responses which can consist of string labels like classes and tags or numeric value. To later predict that correct response when posed with some new examples, the model will obtain certain numeric values or string labels. The supervised approach is very similar to human learning under some kind of supervision like a teacher or like a parent. The teacher or parent provides to students and children some good examples needed to be remembered by children. Further, child or student is able to derive some general rules when it comes to these specific examples.

It should be noted that regression problems target certain numeric values, while classification problems target some qualitative variable like tag or class. A regression task mainly determines that average prices of houses located in Los Angeles, and classification problem may distinguish different types of flows based on their petal and sepal measures.

Unsupervised Learning

When it comes to the unsupervised machine learning algorithms, we do not have any outcome or target variable to estimate or predict. This technique is used mainly for clustering population when it comes to the different groups. The technique is also widely used for segmenting customers in certain groups for particular intervention. Common unsupervised algorithms include K-means and Apriori algorithm. Unsupervised learning occurs when a model or algorithms learn from certain plain examples with no any associated response. It means that the technique leads to the model notably determine by itself the certain data patterns. This type of algorithm also tends to completely restructure the data into something completely different like some new features commonly representing a new series of different uncorrelated values or some new class. These models are of great significance when it comes to the providing humans with valuable insight into the overall meaning of data and providing assistance when it comes to the relevant and useful inputs to other models like supervised learning algorithms.

This learning approach greatly resembles the methods which humans use in order to figure out some certain events or objects within the same class, commonly by observing that degree of similarity existing between objects. A wide range of recommendation system which you can find on the web like marketing automation is mainly based on unsupervised learning techniques. The marketing automation model is able of deriving its suggestion based on your purchase history on based on that what you have bought in the past. These recommendation systems are mainly based on a certain estimation of what group of loyal customers you resemble the most. This system then by inferring your most likely preferences based on that obtained group of customers.

When it comes to the unsupervised learning, certain input data does not have any known result and it is not labeled. An unsupervised algorithm is prepared by firstly deducing certain structures notably present within the input data. This occurs in order to be able to extract some general rules. It may be done by a certain mathematical process in order to systematically reduce that redundancy. The another approach used commonly is to organize data obtained based on similarity. Example unsupervised problems include dimensionality reduction, clustering and association rule learning.

Semi-Supervised Learning

When it comes to the most common machine learning algorithms, I have to mention semi-supervised learning algorithms. In this particular approach, input data represents a certain mixture of some unlabelled examples in addition to labeled examples. In this approach, there is that desired prediction problem, but the algorithm first must learn the structures in order to organize data and to make predictions. Regression and classification can be semi-supervised. Semi-supervised learning algorithms are, in fact, extensions to some other flexible methods notably making an assumption about how to deal with and model the unlabeled data.

When it comes to the crunching data to model different business decision, you will use unsupervised and supervised learning methods. On the other hand, a hot topic, especially in recent times, is semi-supervised learning method when it comes to the different areas like image classification where you have those large datasets commonly containing a very small number of labeled examples.

Reinforcement Learning

This book revolves around the third type of machine learning reinforcement learning which occurs when you have the model containing examples that lack labels, just like unsupervised learning. On the other hand, you are able to accompany any example with some negative or positive feedback in accordance with the solution notably proposed by the algorithm. This technique is commonly connected to different applications when it comes to the making certain decisions by the models. In other words, the product is entirely prescriptive in addition to being descriptive like in unsupervised learning. It should be noted that all decision bear certain consequences. In our human world, it is just like learning new things by error and trial.

In the human world, errors can help us learn because every error has a certain penalty added like loss of time, cost, pain or regret. Errors are teaching us that a specific course of our actions is less likely to turn to success than some other actions. One of the most interesting reinforcements learning models are those integrated into computers which learn to play complex video games with no human intervention. In this particular case, an application presents the model with different examples of several situations, like that situation when a gamer is stuck in a maze due to avoiding an enemy. The application further lets the model know the result of actions notably needed in order to avoid a dangerous situation and avoid an enemy.

When it comes to the reinforcement learning used in video games, applications can learn and discover situation leading to danger and pursue survival. If you are interested in video games and incorporation of reinforcement learning in games, you can have a look at how the leading company in the world, Google DeepMind was able to create a certain reinforcement learning program especially able to play by itself Atari's video games. If you decide to watch the program, you will notice that it is very clumsy at the beginning as well as very unskilled. However, the program is able to steadily improves with more training until eventually, it becomes a champion in the game.

Reinforcement learning approach uses the machine notably trained in order to make a certain decision. It commonly works in the way when the certain machine is exposed to a certain environment where it is able to trains itself continually only using error and trial. This machine further is able to learn from its past experience and it tries to capture that best possible knowledge in order to make the most accurate business decisions. Reinforcement learning is a certain decision process, specific reward system, and recommendation system.

The most common reinforcement learning algorithm is Markov Decision Process which will be explained later in the book. On the other hand, commonly used supervised and unsupervised learning algorithms used include Linear Regression, Logistic Regression, Naive Bayes, Support Vector Machines, Decision Tree, K-Means, Random Forest, KNN, Dimensionality Reduction Algorithms and Gradient Boosting Algorithms like GBM, XGBoost, CatBoost, and LightBoost. However, here we are interested in reinforcement learning and some common reinforcement learning algorithms notably explained in the following chapters. Since you are familiar with common machine learning techniques, it is time to get to know fundamentals of reinforcement learning before we dive into some practical examples of reinforcement learning with Python.

Chapter 2 Elements of Reinforcement Learning

Reinforcement learning is learning about what to or how to map different situations to actions in such a manner to maximize that numeric reward signals. It should be noted that the learning is no told what specifically to do, what to perform and what action to take like in the other machine learning techniques, but instead, it is supposed to discover by itself which actions will most likely yield the most reward.

That idea we were able to learn by interacting with the environment is first to occur to everyone notably when we think about the overall nature and fundamentals of a learning process. Reinforcement learning is very similar to actions when a child learns how to walk when a child waves its arms. In this case, the child has no any explicit teacher, but it does have that direct sensorimotor connection to things and humans in his environment. It should be noted that exercising this environmental connection produces relevant information about the effect and cause as well as about different consequences of different actions. These connections also tell what to do when it comes to the achieving different goals.

Interactions with the environment are undoubtedly a major source of valuable knowledge, about ourselves and about our environment. We are aware of how our environment responds to certain actions performed by us. We further seek in order to influence what occurs through our certain behavior. The foundational idea notably lying nearly all machine learning theories is learning from an interaction. In this book, we simply explore and discover different reinforcement learning techniques and learning from interaction. We will also explore some idealized learning situations as well as evaluate the overall effectiveness of different reinforcement learning techniques.

In most challenging as well as most interesting cases, different actions may affect both the following situations and immediate reward. By doing so, actions are all of affecting all subsequent rewards as well. These characteristics are error and trial search and some delayed reward. It should be noted that all these characteristics are the most important certain distinguishing features when it comes to the reinforcement learning.

It should be noted that reinforcement learning is not defined by certain characterizing method, but it is mainly characterizing a certain learning problem. Any method or technique notably well suited when it comes to the solving problem, we will consider as a reinforcement learning technique. The basic idea lying behind any reinforcement learning problem is to simply capture those most important aspects of some real problem by facing a certain learning agent notably interacting with an environment in order to achieve a certain goal. It should be noted that such a learning agent must be able to properly sense the overall state of its environment into some certain extent. It also must be able to take different actions specifically affecting the overall state. The certain learning agent also must have a particular goal notably related to the overall state of its environment. This certain formulation is intended to focus and include three aspects, goals, aspects and sensation. These three fundamental aspects are included in their simplest forms without doing anything to them in order to trivialize them.

Reinforcement learning differs from supervised learning techniques greatly. It is the kind of machine learning technique notably studied in some most current research when it comes to the statistical pattern recognition, machine learning, and artificial neural networks. On the other hand, supervised learning techniques focus on learning from certain examples notably provided by some knowledgeable supervisor. This is very important part of machine learning, but alone this technique is not entirely adequate when it comes to the learning from certain interactions. In interactive learning like reinforcement learning, problems involved are often very impractical in order to obtain certain examples of that desired behavior which is correct as well as representative of all different situations in that the certain agent has to act and perform different tasks. In reinforcement learning, an agent simply must be able to collect valuable information by itself and learn from its environment by error and trial.

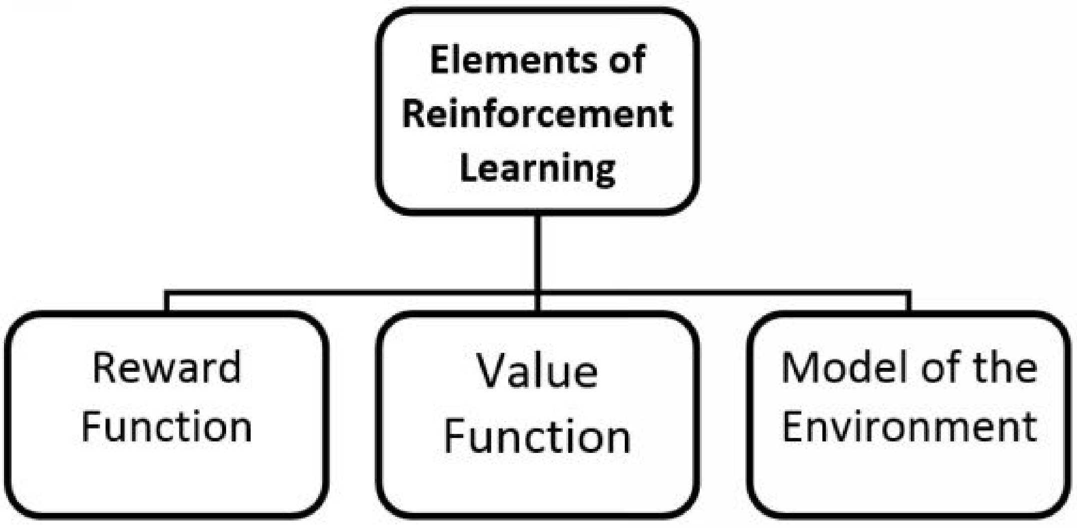
When it comes to the different challenges, which commonly arise in reinforcement learning projects, I have to mention that in this technique, learning is understood as a certain trade-off between exploitation and exploration. In order to obtain a great reward, a certain reinforcement learning agent must prefer certain actions which are has tried in its past and found to be very effective when it comes to the producing a certain reward.

In order to discover suitable actions in order to get a reward, a reinforcement learning agent has to try different actions which have not selected in its past. The agent also has to exploit information that has already know to obtain a certain reward. It also has to explore to make some better action selection in the near future while moving towards a certain reward. It should be noted that neither exploitation and exploration can be pursued without failing at some point, and that is the greatest dilemma when it comes to the reinforcement learning projects. Therefore, a certain learning agent must try different actions in order to progressively favor those which appear to be the most suitable. If we look at a certain project on a stochastic task, then every action must be tried several times in order to gain that reliable estimate of some expected reward.

The exploration-exploitation dilemma, in fact, has been intensively studied by different mathematicians in the past few decades. For now, we are sure that the entire issues when it comes to the finding that balance between exploitation and exploration does not even arise in other learning techniques like supervised learning. However, when it comes to some larger trends of that reinforcement learning is a great part towards that greater interaction and contact between different engineering disciplines and artificial intelligence. It should be noted that not that long time ago, a field of artificial intelligence was viewed as entirely separate from statistics and control theory. However, over the last decades, that view eroded. Therefore, modern artificial intelligence approach accepts control theory as well as statistics.

Before we get to know the essentials and basic elements of reinforcement learning, the best way to understand the concepts lying under reinforcement learning is to get to know some of the most common examples and reinforcement learning applications which greatly guided the overall development of this machine learning approach. You can think of a master chess player who is about to make his move. The overall choice is informed by not anticipating possible replies as well as counterreplies and by planning different moves. The choice is also informed by some immediate as well as intuitive judgments of that great desirability of certain moves and positions.

Another great example of reinforcement learning is an adaptive controller notably label of adjusting parameters of some petroleum refinery's operation situated in real time. The controller is able of optimizing the quality trade-off on the certain basis of already specified marginal costs, but without sticking strictly to that set of points notably suggested by engineers. Another example is a gazelle calf which struggles to its feet just minutes after being born. However, half an hour later, it is able to run approximately at 20 miles per hour. In the following section, you will get to know the basic elements of reinforcement learning.



Beyond the environment and the agent, one model is able to identify four other subelements of common reinforcement learning system including a reward function, a policy, and models of the environment. When it comes to the policy, it is able of defining the certain learning agent'way of specific behaviors at some given time. Generally speaking, a certain policy is a mapping from some perceived states of the certain environment to specific actions notably taken within those certain states. A policy also corresponds to what in certain psychology field would be named a set of certain stimulus-response and associations.

There are also some cases when a policy is a very simple function of some lookup table. On the other hand, in some cases, a policy may commonly involve some extensive computation like a search process. The policy is the basis of a reinforcement learning agent in that common sense when a certain agent is sufficient when it comes to the determining certain behavior. It should be noted that policies can be stochastic as well.

The common reinforcement element is also a reward function. It is able of defining that specific goal in a reinforcement learning issue. Generally speaking, a reward function is able to map every perceived state and state-action pairs of the certain environment to a single number or reward. Further, it indicates that desirability of that certain state. When it comes to the ultimate reinforcement learning objective, it is to maximize that total reward it receives in terms of a long run. The reward function also defines what are bad and what are good events for the reinforcement agent.When it comes to the biological system, it simply would not be proper to identify certain rewards with pain and pleasure since there are many immediate as well as defining features of the issue notably faced by the reinforcement agent.

It should be noted that any reward function must be unalterable by certain reinforcement agent. However, it commonly serves as a basis when it comes to the altering policies. For instance, is some action is selected by a certain policy notably followed by some low reward, that policy may be easily changed in order to select some completely other actions in that certain situation in the near future. It should be noted that reward functions also may be stochastic.

A reward function commonly indicated what is good and what is bad in some immediate sense, while a value function is able of specifying what is good, but in the long run. Generally speaking, the value of a certain state is that total amount or reward an agent may expect to accumulate over sometime in the future, starting from an initial state. Values also are able of indicating that long-term desirability of certain states especially after taking into account different states which are likely to follow as well as to reward agents available in that certain states.

In human analogy, rewards are just like pain and pleasure. Therefore, values commonly correspond to some more refined and more farsighted judgment of how displeased and pleased we are when our environment in some certain state. So, expressed in this way, we simply hope it is very clear that value function will formalize that familiar and basic idea.

It should be noted that reward in a certain sense primarily, while on the other hand, certain predictions of rewards are entirely secondary. Just remember that without rewards there would be no any values, and the main purpose is to estimate values in order to achieve more rewards. In fact, it is a value with that we are commonly most concerned when we are making and evaluating different decisions. Certain action choice is made notably based on some value judgments. We commonly seek different actions which bring some states of the highest value and not states with the highest reward since certain actions are able of obtaining that greatest amount when it comes to the rewards in terms of some longer run. In planning and decision making, that derived quantity is called value, and that is what concerns us the most.

It should be that determining values in the greater challenge than determining rewards since rewards commonly are given directly by the certain environment, while values, on the other hand, must be reestimated and estimated from certain sequences of different observations notably done by an agent noticing its environment and learning from it over its entire lifetime. It should be noted that the most important component of the majority of reinforcement learning models is a method for more efficient estimating values.

Reinforcement learning is all about observing and learning from the environment done with interacting. It is our greatest belief that those certain methods are able to take some advantage of certain details of some individual behavior interactions and that these methods are commonly much more efficient that some other evolutionary cases.

On the other hand, evolutionary methods commonly ignore that much useful structure when it comes to the common reinforcement learning problems. These methods also do not use fact that states that an individual passes through different states during its lifetime. Therefore, in many cases, the relevant and important information may be misleading when these states are misperceived.

Even though reinforcement learning and evolutionary methods have so much in common, we simply do not consider these evolutionary methods by themselves to ve well suited when it comes to the common reinforcement learning problems. The final element of reinforcement learning system in optionally a model of the environment. A model fo the environment simply mimics that certain behavior of the environment.

- Reward function defines the ultimate goal in a reinforcement learning problem. In order to achieve this goal, a policy is altered.
- Value function: Reward function indicate what is good when it comes to the immediate sense while a value function is able of specifying what is good in the long run.
- Model of the environment: Predict and mimic the behaviors of the environment. Model of the environment is commonly used for planning is you know that current state and action. In this

case, it can predict the resultant next stage as well as next reward.

As I already mentioned before, the model of the environment is able of predicting as well as mimicking the certain environment. For instance, given a certain state as well as an action, the model may be able to predict that resultant next stage as well as the next reward. These models are commonly used for planning and for anyway when it comes to the deciding on a certain course of action mainly by considering that possible future situation even before these actions actually were experienced by the reinforcement learning model.

When it comes to the incorporation of these models, I have to say that the models are something newly integrated into reinforcement learning projects. It is fair to say that these models are absolutely new development. When it comes to some early reinforcement learning systems, these were entirely trial-and-error learners. These early systems were entirely opposed to the planning methods. However, techniques have improved since it becomes clear that reinforcement learning is absolutely related to dynamic programming approach and techniques since reinforcement learning uses different models related to state-space planning methods.

Chapter 3 Markov Decision Processes

Markov decision processes or MDPs commonly provide that mathematical framework when it comes to the modeling some decision making especially in situations where certain outcomes are partly random and partly under some control of decision maker. MDPs are very useful when it comes to the studying various optimization problems mainly solved via reinforcement learning and dynamic programming. MDPs are commonly used in different scientific areas like automated controls, robotics, manufacturing, and economics.

Generally speaking, an MDP is a very discrete time stochastic control process which at every step time, the process is in a certain state. Further, the decision maker chooses any action notably available at some certain state The process then responds by moving to the following state notably depending on the current stage as well as depending on the decision maker’s stage. It should be noted that both elements are conditionally independent of any previous actions and states. In other words, that certain state of some MDP satisfies the overall Markov property.

It should be noted that Markov decision processes are, in fact, the big extension of Markov chains. However, these two differ from each other since there is the addition of different actions when it comes to the Markov decision processes. This means that in an MDP there are allowed different rewards as well as different choices notably giving motivation. It should be noted that if there is only a single action for every state, then all rewards will be the same. In this particular case, any Markov decision process will be reduced to a Markov chain.

Markov decision process is commonly a 5-tuple containing a finite set of different states and a finite set of different actions. Any Markov decision process also contains the probability function regarding the probability that some action in a specific state at a certain point in time will lead to another particular state at given time. An MDP also contains certain discount factor notably representing the overall differences existing in the importance between a present and future rewards. The last element of a Markov decision process is certain immediate rewards or some expected immediate reward notably received after certain transitioning from one state to another state due to a specific action.

When it comes to the core problems of MDPs, it is to find that certain policy for the specific decision maker. It is represented by the certain function specifying all actions that certain decision maker will choose when in a certain state. It should be noted that a Markov decision process is commonly combined with a certain policy in a manner that it fixes all actions for every state and the certain state. It leads to the resulting combination notably behaving as a Markov chain. The ultimate goal is to choose a certain policy which will most likely maximize the certain cumulative function of the specific random rewards. It is typically represented as the expected discounted sum over that potentially infinite horizon.

In the following section, you will see Python MDP toolbox with classes and functions in order to create Markov Decision Process. The classes are developed entirely based on the MATLAB MDP toolbox. The first array manipulation is done using NumPy. On the other hand, full sparse matrix support is done using SciPy’s sparse package. There is also an optional linear programming support done using cvxopt. In the following section, you will see eight Markov decision implementation using Python. In order to do it this way, SciPy and NumPy must be installed on your system. If there are no these libraries installed on your system, use their documentation in order to get them.

First, you will install Python’s toolbox PyPI. In order to get one, you just have to type pip install and pump toolbox. As soon as you have it, you will change it to the PyMDP Toolbox directory. You should install the repository via Setup tool, or you will not have needed administrative access. The following step is to clone the Git repository. After you done this you will follow step by step instructions. I assume you already know how to use Git, so that will not be a problem. The next step is to import the module and set up an example MDP problem by using a certain discount value of, let's say 0.9. You will solve it using the value iteration model and further check that optimal policy.

```
import mdptoolbox . example
P, R = mdptoolbox . example . forest ()
vi = mdptoolbox . mdp . ValueIteration ( P, R, 0.9 )
vi . run ()
vi . policy
```

Displaying the Relevant Documentation

```
import mdptoolbox
mdptoolbox? <ENTER>
mdptoolbox . mdp? <ENTER>
mdptoolbox . mdp . ValueIteration? <ENTER>
```

Python Markov decision process toolbox provides function as well as classes when it comes to the resolution of some discrete-time MDP. When it comes to the available modules, the toolbox includes Markov decision process algorithms, a function for working and validating an MDP and several examples of reward matrices and transitions notably forming a valid Markov decision process. When it comes to the using this certain documentation, it is available as both in pdf format and docstrings.

Use the Built-in Examples and Import Example Module

```
import mdptoolbox
import mdptoolbox . example
```

As soon as you imported the examples, it is not necessary to issue that import MDP toolbox command. It should be noted that certain code snippets will be indicated by certain three greater-than signs, so you do not get confused. In order to view doctrines of the certain value iteration of a specific class, you will run value interaction code.

```
x = 17

x = x + 1

x 18
```

When it comes to the available classes in Python MDP toolbox, you will find base MDP data, backward induction of finite horizon Markov decision process. You will also find classes including Q-learning Markov decision process, policy iteration MPD, value iteration MDP, modified policy iteration MDP and Gauss.Seidel value interaction Markov decision process. When it comes to the parameters available, you will find a discount factor, reward vectors and matrices, transitions or arrays, a different number of peri+dios, optional terminal rewards, ship check, attributes or data, optimal value function, optimal policy and used CPU time or time float.

```
import mdptoolbox, mdptoolbox . example

P, R = mdptoolbox . example . forest ()

fh = mdptoolbox . mdp . FiniteHorizon ( P, R, 0.9, 3 )

fh . run ()

fh . V array ([[ 2.6973, 0.81 , 0. , 0. ], [ 5.9373, 3.24 , 1. , 0. ], [ 9.9373, 7.24 , 4. , 0. ]])

fh . policy array ([[ 0, 0, 0 ], [ 0, 0, 1 ], [ 0, 0, 0 ]])
```

You are also able to set the Markov decision process to the silent mode using set silent command. You also can set the Markov decision process to the verbose mode using set verbose command.

```
run ()
```

setSilent ()

setVerbose ()

Markov Decision Processes Parameters

In this section of the book, we will have a look at MDPs parameters including reward, discount, transitions, epsilon, skip check, maximum iteration and other. We have already know that the scope is to find that optimal policy notably a solution which specifies what we are supposed to do for each state other than that ultimate goal. Before we get into Python code for a Markov decision interface, you should get to know some of the most commonly used Markov decision parameters.

- Transitions: Arrays or transitions are different probability matrices. These can be easily defined in different ways. The simplest method is a numpy array, which has the shape of actions. However, there are also some other possibilities like tuple or list of different numpy objects of certain lengths. In this case, each element will contain that numpy array or matrix with the certain shape. This approach comes very usefully when certain transition matrices are, in fact, sparse. It should be noted that every action's transition should be index-able.
- Reward: Reward matrices are also called vectors. Just like the transition matrices, reward matrices also may be defined in different ways. The simplest method again is to form a numpy array with a certain shape. A list of lists may be used as well where every inner list may be composed of a certain outer list with a certain length of different actions. It should be noted that certain outer list may be replaced by some other object which can be indexed as a reward, object array of certain length of actions and numpy object.
- Discount: Discount factor represents certain per time-step discount factor on every future reward. It should be noted that valid values are commonly greater than zero and that they include value one. It the case when discount factor is one, the

convergence cannot be assumed and there will be a certain warning displayed. It should be noted also that specific subclasses of Markov decision process may pass as none in the situation where the model does not use that certain discount factor.

- Epsilon: Epsilon when it comes to the Markov decision processes is certain stopping criterion. Using this parameter, that maximum change in a certain value function and each iteration is compared against certain epsilon function. The value function is considered to have certain coverage to the specific optimal function in the case when the certain change falls below the value. It should be noted that certain subclasses of Markov decision process may easily pass to none in the situation where the model does not use that epsilon-optimal stopping criterion.
- Maximum Iteration: Maximum iteration parameter represents that maximum number of possible iterations. The Markov decision process model will be terminated as soon as this iteration comes at that stage where they simply elapse. This must be greater than zero in the case when specified. It should be noted that certain subclasses of Markov decision process may pass none in the situation when the model does not use that maximum number of possible iterations.
- Skip Check: We run this parameter in order to check on rewards and transitions arguments in order to make sure that they work properly and describe a certain valid Markov decision process. You may set this argument to true to skip this certain MDP check.

A Markov decision process contains:

- A set of possible world states denoted as S
- A set of possible action detoned as A
- A real value reward function denoted as R
- A decsription T of every action's effects in every state

Representing actions:

- Deterministic Actions: For every state and action we need to specify a entirely new state.
- Stochastic Actions: For every state and action we need to specify a probability distribution over following states representing the certain distribution P(s' | s, a)

Some other common Markov decision parameters include policy zero which is a certain initial policy, attributes or data, eval type of the function used in order to evaluate that zero matrix in order to solve as a certain set of linear equations, tuple or the optimal value functions where every element is a certain effort corresponding to some expected value of being in certain state assuming that optimal policy is followed.

Other parameters used include time notably used to converge to some optimal policy, verbose, iteration showing CPU time, average time showing the average reward when it comes to the certain optimal policy, initial value showing the staring value function as vector of zeros and iter representing the total number of iterations notably taken in order to complete the computation. In the following section, you will get to know the code for a Markov decision process interface, the code used for the modified dict class and code for the Toy Markov decision process.

Markov Decision Process Interface

```
class MarkovDecisionProcess
def transition ( self, from _ state, action, to _ state )
raise NotImplementedError def initial _ state ( self )
raise NotImplementedError def reward ( self, state )
raise NotImplementedError def discount ( self, state )
raise NotImplementedError
```

Modified Dict Class

```
class SumDict ( dict )
def __ setitem __ ( self, key, value )
self . has _ key ( key )
value += self . get ( key ) dict . __ setitem __ ( self, key, value )
```

Toy Markov Decision Process

```
import numpy as np
class ToyMDP ( MarkovDecisionProcess )
def __ init __ ( self )
self . world = np.array ([ [ -0.04, -0.04, -0.04, 1 ], [ -0.04, None,
-0.04, -1 ], [-0.04, -0.04, -0.04, -0.04 ], ])
self . initial _ state = ( 0, 0 )
self . finals = [( 0,3 ), ( 1,3 )]
self . actions = ( ‘l’, ‘r’, ‘u’, ‘d’ )
def __iter __ ( self )
def __init__( self, iterator, finals )
self . iterator = iterator
self . finals = finals def next ( self )
coords = self . iterator . coords
val = self . iterator . next ()
return Iterator ( self . world . flat , self . finals )
def _move ( self, state, action )
shape = self . world . shape
next = list ( state )
```

```
elif action = = ‘l’
self . world [ state [0]] [ state [1]-1] ! = None )
next [1] - = 1
return tuple ( next )
def successors (self, state, action
d = SumDict ()
action = = ‘1
d[self ._ move ( state, ‘l’)] = 0.8
d[self ._ move ( state, ‘u’ )] = 0.1
d[self ._ move ( state, ‘d’ )] = 0.1
return d
def transition ( self, from _ state, action, to _ state )
return self . successors ( from _ state, action) [ to_state ] def initial
_ state (self)
return self . initial _ state
def reward ( self, state )
return self . world [ state [0]] [ state [1]]
def discount ( self )
return 1
```

When it comes to some probabilities where you have unknown rewards, Markov decision process is entirely a reinforcement learning problem. For this certain purpose, it is very useful if you use a method when you firstly defines a certain further function that corresponds to taking specific action and further continuing that optimally actions. It should be noted that in the certain case, the function would be unknown as well, but that experience during learning will be based on a set of actions and set of states put together with the certain outcomes which are state and actions being tried together at the same time. This is named Q-learning and in the following chapter, you will get to know this certain approach as well.

It should be noted that reinforcement learning could solve various Markov decision processes without any explicit specification when it comes to the different transitions probabilities. In this case, the values of different transition probabilities are only needed in policy and value iteration.

Chapter 4 Approximate Dynamic Programming

In mathematics, computer science, economics, bioinformatics and many other scientific fields, dynamic programming is widely used in order to solve some complex problems mainly by breaking a problem down into a certain collection of some simpler problems. Dynamic programming further leads to solving there simpler problems by solving each of them once and then storing their overall solutions. Therefore, dynamic programming solves each problem instead of just re-computing their solutions. Using this approach, the models look up the some previously computed solution, notably leading to less computation time at the expense of some modest expenditure in storage space.

It should be noted that every sub-problem solution is mainly indexed in a certain manner mainly based on the certain values of its initial parameters. The technique of storing different solutions to some sub-problems instead of just re-computing them is named memorization. It should be noted that dynamic programming is also known as dynamic optimization. Algorithms obtained by this method are commonly used for optimization. A model is able to examine that previously solved sub-problems and it further combines their solutions in order to give the best solution for that given problem.

There are greedy algorithms as well that treat the solution as a sequence of certain steps and further picks that locally optimal choice at every step, which differs from dynamic algorithms. A greedy algorithm also does not guarantee that optimal solution, since picking some locally optimal choices commonly result in some bad global solution, but it is commonly faster to calculate. On the other hand, greedy models like Prim and Kruskal have proven to lead to that optimal solution.

Dynamic programming is also used when it comes to the counting the number of all solutions in addition to finding that optimal solution. For instance, counting the number of different ways a specific amount of changes may be made from an obtained collection of coins will be estimated using dynamic programming. Also counting the number of possible optimal solutions to the same coin problem will be estimated by dynamic programming.

Dynamic programming in addition to reinforcement learning is commonly used when it comes to the addressing different problems from a variety of fields including operations research, artificial intelligence, automatic control, and economy. Many problems in this specific field are commonly described by different continuous variables where reinforcement learning and dynamic programming can find that exact solution when it comes to some discrete cases. So, when it comes to the practical dynamic programming and reinforcement learning, an approximation is essential.

Dynamic Programming:

- Partitions a problem into overapping sub-problems
- Stores solutions of sub-problmes and avoids calculations of same quality twice
- Soecific mainly bottom up learning models in that manner that the smallest sub-problems are explicitly solved first and the results of these are used in order to construct solutions to progressively larger sub-instances

In the following section, you will see how to incorporate dynamic programming in order to get policy evaluation, value iteration, and policy iteration. Like I already mentioned Markov decision process problems can be solved using dynamic programming when we suppose that there is the state function denotes as P and certain rewards function denoted as R. When we wish to calculate the certain policy which maximizes that expected discounted reward we will use dynamic programming. The common family of models in order to calculate this policy require storage fo certain two arrays of some indexed value denoted as V. It should be noted that this value contains some real values and specific policy contains certain actions. At the end of this algorithm, you will see outcomes containing the solution V notably containing that discounted sum of different rewards specifically to be earned on average by simply following that solution obtained from a set of all states.

Policy Evaluation

In this section of the book, you will see codes for policy evaluation using dynamic programming in Python. You will evaluate a certain policy when you gave already given environment as well as a full description of the overall environment's dynamics. In this case, the policy contains set of actions and set of states where shaped matrix will be representing this policy.

Running command evaluation, you will see the representation of all transition probabilities of certain environments. You will also stop the evaluation once your values function change come to a stage of being less than theta function. The discount factor is represented with a lambda function.

```
import numpy as np
import pprint
import sys
rom lib . envs . gridworld import GridworldEnv
pp = pprint . PrettyPrinter ( indent=2 )
env = GridworldEnv ()
def policy _ eval ( policy, env, discount _factor = 1.0, theta =
0.00001 )
V = np . zeros ( env.nS )
delta = 0
v = 0
```

You will perform a complete backup following the each state. You will start with that random or all zero value functions. The following step is to look at all possible following actions. Then you will calculate that expected values and o the every state you will look at the possible following states.

The following step is to calculate how much value function changed across every state. You will stop evaluating once your value function change to be below that threshold of theta when the delta is less than theta.

```
v += action _prob * prob * ( reward + discount _factor * V [ next_state ])
delta = max ( delta, np . abs ( v – V [ s ] ))
V [ s ] = v
return np . array ( V )
random _policy = np . ones ([ env.nS, env.nA ]) / env . nA
v = policy _ eval ( random_policy, env )
print ( “Value Function:” )
print ( v )
print (””)
print ( “Reshaped Grid Value Function:” )
print (v . reshape ( env . shape ))
print (””)
```

```
Value Function:
[ 0. -13.99993529 -19.99990698 -21.99989761 -13.99993529
-17.9999206 -19.99991379 -19.99991477 -19.99990698
-19.99991379 -17.99992725 -13.99994569 -21.99989761
-19.99991477 -13.99994569 0. ]
Reshaped Grid Value Function: [[ 0. -13.99993529 -19.99990698
-21.99989761] [-13.99993529 -17.9999206 -19.99991379
-19.99991477] [-19.99990698 -19.99991379 -17.99992725
-13.99994569] [-21.99989761 -19.99991477 -13.99994569 0. ]]
```

The last step is testing. You should also make sure that the evaluated policy is matching that expected policy.

Policy Iteration

In this section, you will see how to evaluate policy iteration using dynamic programming in Python. In this case, just as in the previous one, you will evaluate a policy when you have given a certain environment as well as that full description of the certain environment. In this case, a set of actions as well as a set of states will be representing the overall policy just like in policy evaluation. We will use command return representing a vector of a certain length and estimation command representing that value function. You will start with a random or all zero function like in the previous example.

```
import numpy as np
import pprint
import sys if
from lib . envs . gridworld import GridworldEnv
pp = pprint . PrettyPrinter ( indent=2 )
```

The following step is taken a form that policy evaluation example. You will also perform a full backup and look at the all possible following actions. Then you will see how much certain value function changed across every state. Further, you will calculate that expected value and stop evaluating as soon as your value function change is below a particular threshold where delta is less than theta.

```
V = np . zeros ( env.nS )
delta = 0
v = 0
```

```
v += action _prob * prob * ( reward + discount _factor * V [
next_state ])
delta = max ( delta, np . abs (v – V [ s ]))
return np . array (V)
def policy _ improvement (env, policy _ eval _fn = policy _ eval,
discount _factor = 1.0 )
```

The following steps include policy improvement model. The codes iteratively evaluate as well as improve a certain policy until you find that optimal policy. Policy evaluation function is used and it takes three arguments including, evaluation, a certain discount factor, and policy. In this case, a certain discount factor is represented by lambda factor. You will also use return arguments as soon as you have a tuple policy containing V and policy. In this case, a policy is that optimal policy while a matrix of shape set of states and set of actions represents a certain situation where every set of states contains that valid probability distribution over different actions. In this case, V represent that value function for that optimal policy.

Further, you will start with a certain random policy and evaluate that current policy. Then you will be able to set false if you make changes to evaluate policy. The best action will be taken under that current policy. Further, you will find the best action using one-step Lookahead. You will notice that ties are resolved entirely arbitrarily. Then you will greedily update the current policy. In the case when the policy is entirely stable, you have found that optimal policy. On the other hand, you have to run return argument. The last step is to test the value function.

```
policy = np . ones ([ env.nS, env.nA ]) / env . nA
V = policy _ eval _fn ( policy, env, discount _factor )
policy policy _ stable
chosen _ a = np . argmax ( policy [ s ] )
action _ values = np . zeros ( env . nA )
action _ values[a] += prob * ( reward + discount _factor * V
[next _ state ]) best_a = np . argmax ( action _ values )

policy _ stable = False

policy [s] = np . eye ( env . nA)[ best _ a ]

policy _ stable

return policy

v = policy _ improvement ( env )

print ( “Policy Probability Distribution:” )

print (policy)

print (””)

print ( “Reshaped Grid Policy ( 0=up, 1=right, 2=down, 3=left )
:”)

print (np . reshape (np . argmax (policy, axis=1), env . shape))

print (””)

print ( “Value Function:” )

print (v)

print (””)

print ( “Reshaped Grid Value Function:” )

print ( v.reshape(env.shape ))

print (””)

Policy Probability Distribution: [[ 1. 0. 0. 0.] [ 0. 0. 0. 1.] [ 0. 0. 0.
1.] [ 0. 0. 1. 0.] [ 1. 0. 0. 0.] [ 1. 0. 0. 0.] [ 1. 0. 0. 0.] [ 0. 0. 1. 0.] [
1. 0. 0. 0.] [ 1. 0. 0. 0.] [ 0. 1. 0. 0.] [ 0. 0. 1. 0.] [ 1. 0. 0. 0.] [ 0. 1.
0. 0.] [ 0. 1. 0. 0.] [ 1. 0. 0. 0.]]

Reshaped Grid Policy ( 0=up, 1=right, 2=down, 3=left ): [[ 0 3 3 2
] [0 0 0 2] [0 0 1 2] [0 1 1 0]]

Value Function: [ 0. -1. -2. -3. -1. -2. -3. -2. -2. -3. -2. -1. -3. -2. -1.
0.]

Reshaped Grid Value Function: [[ 0. -1. -2. -3.] [-1. -2. -3. -2.] [-2.
-3. -2. -1.] [-3. -2. -1. 0.]]

Expected _ v = np . array ([ 0, -1, -2, -3, -1, -2, -3, -2, -2, -3, -2, -1,
-3, -2, -1, 0 ]) np . testing . assert _ array _ almost _ equal ( v,
expected _ v, decimal = 2 )
```

Value Iteration

In order to get value iteration using dynamic programming, you will also have OpenAI environment. In this case, you will have the transition probabilities of an imported environment. You will use theta as stopping threshold. In the case when the value of every state changes to be less than theta, then you are done. The discount factor is represented in a lambda function. You will also use a helper function in order to calculate the value for every action in a certain state. State argument will be used in addition to the value notably used as an estimator with a vector of a certain length.

```
import numpy as np
import pprint
import sys
sys.path.append (“../”)
from lib . envs . gridworld import GridworldEnv
pp = pprint . PrettyPrinter ( indent=2 )
env = GridworldEnv ()
def value _ iteration ( env, theta = 0.0001, discount _factor = 1.0 )
def one _ step _ lookahead ( state, V )
A = np . zeros ( env . nA )
A[a] += prob * ( reward + discount _factor * V [ next _ state ])
return A
```

The following step is to use stopping condition and update every state. You will also do a one-step lookahead in order to find the best possible actions. Further, you will calculate delta across every state notably seen so far and update that value function. The following is to perform checking if you can stop. Further, you will create a deterministic policy by using that optimal value function.

```
V = np . zeros ( env.nS )
delta = 0
A = one _ step _ lookahead (s, V)
Best _ action _ value = np . max (A)
delta = max (delta, np . abs( best _ action _ value – V [s]))
V[s] = best _ action _ value
delta < theta
break
policy = np . zeros ([ env . nS, env . nA ])
A = one _ step _ lookahead (s, V) best _ action = np . argmax (A)
Policy [s, best _ action ] = 1.0
return policy, V
policy, v = value _ iteration ( env )
print ( "Policy Probability Distribution:" )
print (policy)
print ("")
print ( "Reshaped Grid Policy (0=up, 1=right, 2=down, 3=left):")
print (np . reshape(np . argmax(policy, axis=1), env . shape))
print ("")
print ("Value Function:")
print (v)
print ("")
print ("Reshaped Grid Value Function:")
print (v. reshape( env . shape ))
print ("")
```

Chapter 5 Integrating with OpenAI Gym

In this chapter, you will see how to integrate with OpenAI Gym. In the previous section, we have used OpenAI Gym in order to evaluate certain reinforcement learning parameters like value iteration and policy evaluation. OpenAI Gym has commonly used toolkit when it comes to the developing as well as comparing different reinforcement learning algorithms.

This toolkit makes no assumptions about the overall structure of a certain agent but it makes it compatible with various numerical computation libraries like Theano and TensorFlow. You will be able to use it directly from Python code as well as from other languages.

OpenAI Gym consists of two main parts, the service, and rich open-source library. The open-source library contains a set of numerous test problems as well as different environments which you can use in order to work out any of yours reinforcement learning models. These OpenAI Gym environments have that shared interface notably allowing you to write some general algorithms. On the other hand, OpenAI Gym service contains a site as well as an API notably allowing people to very meaningfully compare overall performance when it comes to their trained agents.

In order to get started, you need Python 3.5 or Python 2.7. You will fetch certain gym code. Later, you will run and install the toolkit. In order to do so, you will need recent pip version. You will be able to work on any package available.

```
git clone
cd gym
pip install -e . # minimal install
```

The following step is to run an environment from OpenAI Gym. You can use an environment for thousand time-steps. Further, you will render obtained environment following the every step.

```
import gym
env = gym . make (‘CartPole-v0’)
env . reset ()
env . render ()
env . step (env . action _ space . sample ())
```

Normally, you will end the simulation even before the cart-pole is allowed to entirely go off-screen. If you are interested in seeing some other environments form OpenAI Gym in action, then replace that cart-pole from the above with something else like MsPacman-v0 or MountainCr-v0. It should be noted that these two will require the Atari dependency. You also may use others as well like Hopper-v1 notably requiring the MuJoCo dependencies. It should be noted that all environments are descent from that base class.

It should be noted, if you are missing any dependencies, you are supposed to get that helpful error message notably telling you what you are missing. When it comes to the installing some missing dependencies the process is very simple. You will only need that MuJoCo license for certain Hopper-v1. The following step is to focus on observations.

If you ever want to perform better in order to take some random actions at every step, you will need to get to know what your actions are doing to certain environments. In order to do so, you have to run step function notably returning exactly what you need. You will get four values including reward, done, info and observation.

When you run step function, it will return value observation. It is entirely environment-specific object notably representing your observation when it comes to the certain environment like certain pixel data from a camera or joint velocities of a robot. On the other hand, you will also get reward function return representing an amount of reward specifically achieved by those previous actions. It should be noted that the scale would vary between different environments. However, the goal of increasing that total rewards always remains the same.

Another valuable function is done or Boolean used when it is the right time to reset that certain environment. Majority of tasks will be divided into some well-defined episodes like true and done. In this situation, true will indicate that the certain episode has terminated. Info function is that diagnostic information that comes very useful when it comes to the debugging. It can be used also for learning as well. On the other hand, official evaluations are not allowed to use this function in this specific learning problem.

The overall process will start when you call that reset function returning an initial observation. It should be noted that writing from that previous code would respect that done flag.

```
import gym
env = gym . make (‘CartPole-v0’)
observation = env . reset ()
env . render () print( observation )
action = env . action _ space.sample ()
observation, reward, done, info = env . step ( action )
print ( “Episode finished after {} timesteps” . format ( t+1 ))
break
```

As soon as you done and hit the break, you will have output where you are able to see where certain resets happened. It will look like this.

```
[-0.061586 -0.75893141 0.05793238 1.15547541]
[-0.07676463 -0.95475889 0.08104189 1.46574644]
[-0.0958598 -1.15077434 0.11035682 1.78260485]
[-0.11887529 -0.95705275 0.14600892 1.5261692]
[-0.13801635 -0.7639636 0.1765323 1.28239155]
[-0.15329562 -0.57147373 0.20218013 1.04977545]
Episode finished after exactly 14 time-steps
[-0.02786724 0.00361763 -0.03938967 -0.01611184]
[-0.02779488 -0.19091794 -0.03971191 0.26388759]
[-0.03161324 0.00474768 -0.03443415 -0.04105167]
```

In the previous example, you have been sampling that random actions form the certain environment's actions space. It should be noted that each environment comes with that first-class space very specific objects notably describing that valid observations and actions.

```
import gym
env = gym . make ('CartPole-v0')
print ( env . action _ space )
#> Discrete (2) print ( env . observation _ space )
#> Box (4,)
```

These discrete spaces will allow some fixed range when it comes to the non-negative numbers. In this certain case, a valid action will be either zero or one. You also have the box space notably representing that n-dimensional box. Therefore, valid observations, in fact, will come as an array containing four numbers. You are able to check that box bounds as well.

```
Print ( env . observation _ space . high)
#> array([ 2.4 , inf, 0.20943951, inf] )
```

```
Print ( env . observation _ space . low )
#> array ([-2.4 , -inf, -0.20943951, -inf])
```

This certain introspection comes as very useful in order to write some generic code for numerous different environments. The most common spaces are discrete and box functions. You are able to sample that space as well as to check that something belongs to that certain space.

```
from gym import spaces
space = spaces . Discrete (8) # Set with 8 elements {0, 1, 2, ..., 7}
x = space . sample ()
assert space . contains (x)
assert space . n = = 8
```

It should be noted that for CartPole-v0 will be one of the actions notably applying great force to the left. On the other hand, some of them will apply force to the right as well. It should be noted that the better your learning algorithm, the less you will have to try to interpret that numbers yourself. When it comes to the environment, the main purpose is to provide that large collection of different environments which expose that common interface that allow algorithmic comparison.

```
from gym import envs
print ( envs.registry . all ())
#> [ EnvSpec (DoubleDunk-v0 ), EnvSpec
(InvertedDoublePendulum-v0), EnvSpec (BeamRider-v0), EnvSpec
(Phoenix-ram-v0), EnvSpec (Asterix-v0), EnvSpec (TimePilot-v0),
EnvSpec (Alien-v0), EnvSpec (Robotank-ram-v0), EnvSpec
(CartPole-v0), EnvSpec (Berzerk-v0), EnvSpec (Berzerk-ram-v0),
EnvSpec (Gopher-ram-v0)
```

This will give you that list of all spaces. Further, these will define that parameters needed for certain task, including that number of total trials required in order to run as well as to maximize the number of all steps. For instance, Hopper-v1 will define that environment where the main goal is to get that two-dimensional simulated robot. It should be noted that these environments are commonly treated as some opaque strings.

To ensure that valid comparison for some future environments, you will not change environments in a manner, which affects the overall performance. When it comes to the adding your own environment to the registry, you will run register command.

Q-Learning Algorithms

In this section of the book, you will see how to explore a family of reinforcement learning algorithms called Q-learning models. It should be noted that these are a little bit different than other mainly policy-based models. Instead of starting with that complex as well as unwieldy deep neural network, you will begin by initially implementing that simple-lookup table version of the models.

The basic gist of every Q-learning model is to have a representation of different environmental states in addition to all possible actions in that states. You will learn the value of every action coming from the each state. This value q, in fact, is referred to that state-action value. In Q-learning, you will start by setting that state-action value to zero. You will go around as well as explore that certain state-action space.

The following step is to try certain action in a state, and you will evaluate that state. In the situation where a state leads to some undesirable outcome, you will reduce that Q value also known as a weight to the value of that action from a specific state.

Therefore, all other actions will have common greater value and they may be chosen the following time when you are examining that particular state. In the case when you are rewarded for taking a certain action, the overall weight of that action for that certain state will increase, so it is more likely that you will choose that state again.

It is important to remember when you update Q, you are also at the same time updating that previous state-action combination. It should be noted that you are able to update Q only after you have seen the results. In the following section, you will see how to implement Q-learning in Python.

```
import numpy as np
import matplotlib . pyplot as plt
from matplotlib . collections import LineCollection
```

The following step is to reward or connect certain graph. Then you will update state, the following state, action, gamma, and alpha. The following step is to renormalize that row to be between zero and one. You will also show all the greedy transversals and then cut off that final arrow.

```
r = np . array ([[-1, -1, -1, -1, 0, -1],
[-1, -1, -1, 0, -1, 100],
[-1, -1, -1, 0, -1, -1],
[-1, 0, 0, -1, 0, -1],
[0, -1, -1, 0, -1, 100],
[-1, 0, -1, -1, 0, 100]]) . astype (“float32”)
q = np . zeros _ like (r)
def update _ q ( state, next _ state, action, alpha, gamma )
rsa = r [state, action]
qsa = q [ state, action ]
```

```
new _ q = qsa + alpha * (rsa + gamma * max(q[ next _ state, :]) -
qsa)
q [state, action] = new _ q
rn = q [state] [q[state] > 0] / np . sum ( q[state][q[state] > 0] )
q [state] [q[state] > 0] = rn
return r [state, action]
def show _ traverse
range ( len ( q ))
current _ state = i
traverse = "%i -> " % current _ state
n _ steps = 0
next _ state = np . argmax (q[ current _ state ])
current _ state = next _ state
traverse += "%i -> " % current _ state
n _ steps = n _ steps + 1
traverse = traverse [:-4]
print ("Greedy traversal for starting state %i" % i)
print (traverse)
print ("")
```

The following step is to show all used or valid transitions. You will also invert that y axis for display. Then you will bump values for viz.

```
def show _ q ()
coords = np . array ([[2, 2],
[4, 2],
[5, 3],
[4, 4],
[2, 4],
[5, 2]])
Cords [:, 1] = max (cords [:, 1]) – cords [:, 1]
Plt. Figure (1, facecolor=‘w’, figsize= (10, 8))
Plt . clf()
ax = plt . axes( [0., 0., 1., 1.] )
plt . axis (‘off’)
plt . scatter (cords [:, 0], cords [:, 1], c=‘r’)
start _ idx, end_idx = np . where (q > 0)
segments = [[ coords[start], cords [stop]]
values = np . array (q[q > 0])
values = values
lc = LineCollection (segments, zorder=0, cmap=plt.cm.hot_r)
lc . set _ array (values)
ax . add _ collection (lc)
verticalalignment = ‘top’
horizontalalignment = ‘left’
x = cords [i][0]
y = cords [i][1]
name = str (i)
y = y - .05
x = x + .05
elif i = = 3
y = y - .05
x = x + .05
elif i = = 4
y = y - .05
x = x + .05
y = y + .05
x = x + .05
```

```
plt . text (x, y, name, size=10
horizontalalignment = horizontalalignment
verticalalignment = verticalalignment
bbox = dict ( facecolor='w', edgecolor=plt.cm.spectral(float( len(
cords )))
alpha=.6)) plt . show ()
```

The core algorithm will need that uncomment in order to see relevant plots within each monitoring. You will also show epsilon-greedy, show transverse and show Q. You should remember not to allow invalid moves at the very beginning and just take some random move. It should be noted that the goal state has reward hundred. The core algorithm will contain gamma, epsilon, alpha, a number of total states, actions and episodes.

```
gamma = 0.8
alpha = 1.
n _ episodes = 1E3
n _ states = 6
n _ actions = 6
epsilon = 0.05
random _ state = np . random . RandomState (1999)
states = list (range (n _ states))
random _ state . shuffle (states)
current _ state = states [0]
goal = False
valid _ moves = r [current _ state]
random _ state . rand ()
actions = np . array (list(range(n _ actions)))
actions = actions
```

```
[valid _ moves = = True]
actions = [actions]
action = actions[0]
next _ state =
np . sum(q[ current _ state ])
action = np . argmax ( q[current _ state] )
actions = np . array (list(range(n _ actions)))
actions = actions[valid _ moves = = True]
random _ state . shuffle (actions)
action = actions [0]
next _ state = action
reward = update _ q (current _ state, next _ state, action,
alpha=alpha, gamma=gamma)
current _ state = next _ state
print (q)
show _ traverse ()
show _ q ()
```

Chapter 6 Monte Carlo Methods

Monte Carlo methods are commonly used when it comes to the model which mimics certain policy iteration. Policy iteration commonly consists of two main steps including policy improvement and policy iteration.

Monte Carlo methods are used commonly in a stage of policy evaluation. In this certain stage, there is a given stationary or deterministic policy and the overall goal is to compute different function values out to find some good approximation to them in terms of all state-action pars.

In this case, we are assuming that the Markov Decision Process is finite and that there is that sufficient memory notably available in order to accommodate that action-values and that the certain problem is entirely episodic. It should be noted that after every episode the new starts from the random initial state. Further, the estimate of the specific value of already given state-action par can be easily computed by averaging that sample returns notably originating from a set of actions and set of states over a period of time.

When you have given sufficient time, this process may lead to constructing some precise estimate Q of the certain action-value function. Further, this finishes that description of that policy evaluation step.

When it comes to the policy improvement step in some standard policy iteration model, the following policy is commonly obtained by computing that greedy policy in respect to Q. When you have given a set of states, this policy will return an action which maximizes Q containing a set of states and set of actions.

In practice, we call this a lazy evaluation notably deferring the computation when it comes to the maximizing different actions to when they are required. It should be noted that this certain method works only in certain episodic problems and infinite and small Markov Decision Processes.

Monte Carlo Prediction

In this section of the book, you will see how to integrate Monte Carlo prediction. The algorithm will be able to calculate the value function for some given policy using sampling. The policy will represent a function which maps an observation to a specific action probabilities. Here, we will also use OpenAI Gym in order to get needed environments. You will also number episodes needed to sample and lambda will represent a discount factor. It should be noted that value is a float while the state is a tuple.

```
import gym
import matplotlib
import numpy as np
import sys
from collections import defaultdict
from lib . envs . blackjack import BlackjackEnv
from lib import plotting
matplotlib . style . use (‘ggplot’)
env = BlackjackEnv ()
def mc _prediction ( policy, env, num_episodes,
discount_factor=1.0 )
returns _ sum = defaultdict (float)
returns _ count = defaultdict (float)
V = defaultdict (float)
Print (“\rEpisode {}/{}.”.format(i_episode, num_episodes),
end=””)
```

```
Sys . stdout . flush ()
```

The following step is to keep track of count and sum of returns for every state in order to calculate an average. You will use an array in order to save all returns. The next step is to get the final value function. Then, you will print every episode which comes very useful for debugging. Further, you will generate an episode. It should be noted that an episode, in fact, is an assay including reward, state, and action.

```
episode = [ ]
state = env . reset
action = policy (state)
next _ state, reward, done, _ = env . step (action)
episode . append ((state, action, reward))
state = next _ state
first _ occurence _ idx
G = sum ([x[2]*(discount _factor**i)
Returns _ sum [state] += G
Returns _ count [state] += 1.0
V [state] = returns _ sum [state]
Returns _ count [state]
return V
def sample _ policy (observation)
V _ 10k = mc _prediction ( sample_policy, env,
num_episodes=10000 )
Plotting . plot _ value _function (V _ 10k, title="10,000 Steps" )
V _ 500k = mc _prediction (sample _ policy, env, num _
episodes=500000)
Plotting . plot _ value _function (V_500k, title="500,000 Steps")
```

The next step is to find all states notably visited in this certain episode. You will also convert every state to a tuple so you can use it as a dict key. Further, you will find that first occurrence as a certain state in that episode and sum all reward since that first occurrence. The next step is to calculate that average return for this certain state over all sampled states.

Monte Carlo Tree Search

When it comes to the subject of game artificial intelligence, it all begins with that perfect information. These, in fact, are turn-based games where players have no that information hidden from other players, so there is no that element of chance when it comes to the game mechanics.

In this types of games everything is fully determined, so Monte Carlo tree can be easily constructed containing all possible outcomes as well as a value notably assigned corresponding to a loss or a win for players. Finding that possible play is a matter of doing a certain search on a tree. It is a certain method of choice and picking that maximum and minimum value. The algorithm is called Minimax.

In order to incorporate simple Monte Carlo tree search in Python, you will need Board class that has a purpose of encapsulating the rules of the certain game. Board class only cares about the artificial intelligence model rather than being concerned with Monte Carlo class. It obtains information about the certain game.

```
class Board (object)
def start (self)
pass
def current _player (self, state)
pass
```

```
def next _ state (self, state, play)
pass
def legal _ plays (self, state _ history)
pass def winner (self, state _ history)
pass
```

It should be noted that you will require certain state data structure containing equivalent states which have that same value. You can use flat tuples as your state data structure. Certain AI class will be constructed using certain interface.

```
class MonteCarlo (object)
def __init__ (self, board, **kwargs)
pass
def update (self, state)
pass
def get _ play  (self)
pass
def run _ simulation (self)
pass
class MonteCarlo (object)
def __init__ (self, board, **kwargs)
self . board = board
self . states = [ ] def update (self, state)
self . states . append (state)
```

The next step is to move towards bookkeeping and initialization. The board object is the place where AI will start obtaining some information about the game. Therefore, you will need to store that information as well as to keep track of certain state data as you get it.

Chapter 7 Temporal Difference Learning

Temporal Difference methods or learning is certain prediction based machine learning approach. It is mainly used for reinforcement learning.The approach, in fact, is a certain combination of Monte Carlo methods and dynamic programming. Temporal difference greatly resembles Monte Carlo methods since it also allows learning by sampling certain environments according to the specific policy. O the other hand, the temporal difference is related to dynamic programming since is also approximates some of its current estimate notably based on some previously learned estimates specifically known as the process of bootstrapping.

The temporal different model is commonly related to temporal difference model when it comes to the animal learning. This is a certain prediction method since temporal difference learning considers that numerous subsequent predictions to be related in some manner. When it comes to the supervised predictive learning, an agent is able to learn form observed values and prediction is made when a certain observation is available. On the other hand, overall prediction mechanism is further adjusted in order to better match a certain observation. Temporal difference learning is all about adjusting predictions to match other more accurate prediction regarding the future. In the following section, you will see how to integrate Q-learning solution using temporal difference method in Python. In order to start, you will cerate an epsilon-greedy policy notably based on some given q-function and epsilon. You will use a Q dictionary that maps from a certain state to certain action-values. It should be noted that every value represents a certain numpy array of a certain length. Epsilon is a function representing that probability to select some random actions. The following step is to get a total number of actions in the certain environment.

% matplotlib inline

import gym

import itertools

```
import matplotlib
import numpy as np
import pandas as pd import sys
sys . path . append (“../”)
from collections import defaultdict
from lib . envs . cliff _ walking import CliffWalkingEnv
from lib import plotting
matplotlib . style . use (‘ggplot’)
env = CliffWalkingEnv ()
def make _ epsilon _ greedy _ policy ( Q, epsilon, nA )
def policy _fn (observation)
A = np . ones (nA, dtype=float) * epsilon / nA
Best _ action = np . argmax (Q[observation])
A [best _ action] += (1.0 - epsilon)
return A
return policy _fn
```

The Q-learning model is entirely off-policy temporal difference able of finding that optimal greedy policy while following that epsilon-greedy policy. The following step is to find the total number of episode notably needed to run for and to find that lambda time discount factor. Alpha will determine a certain learning rate while epsilon represents that chance when it comes to the sampling some random actions. It should be noted that a float will be between zero and one.

```
def q _ learning ( env, num _ episodes, discount _factor=1.0,
alpha=0.5, epsilon=0.1 )
Q = defaultdict ( lambda: np . zeros (env . action _ space.n ))
stats = plotting . EpisodeStats
( episode _ lengths=np.zeros (num _ episodes)
Episode _ rewards=np . zeros (num _ episodes))
policy = make _ epsilon _ greedy _policy ( Q, epsilon, env.action _
space.n )
sys . stdout . flush ()
state = env . reset ()
action _probs = policy (state)
action = np . random . choice (np.arange (len(action_probs)),
p=action_probs)
next _ state, reward, done, _ = env . step (action)
stats . episode _ rewards [i_episode] += reward
stats . episode _lengths [i_episode] = t
best _ next _ action = np . argmax (Q[next_state])
td _ target = reward + discount _factor * Q [next_state]
[best_next_action]
td _ delta = td _ target – Q [state][action]
Q [state] [action] += alpha * td _ delta
break
state = next _ state
return Q, stats
Q, stats = q _ learning (env, 500)
Episode 500/500
Plotting . plot _ episode _ stats (stats)
```

In this certain temporal learning example, you will get that action-value function. You will also obtain that nested dictionary notably mapping states including action and action-value. It should be noted that you have to keep track of all useful statistics like episode lengths and episode rewards. The next step is to print out that episode notably very useful for debugging and to update TD.

Conclusion

Reinforcement learning is a wide area of machine learning mainly inspired by another scientific field, behaviorist psychology. Reinforcement learning is commonly concerned with how certain software agents can take different actions in different environments in order to maximise that notion when it comes to the cumulative reward. Reinforcement learning is also one of the most active areas when it comes to the artificial intelligence since its computational approach is to focus on whereby a reinforcement agent acts toward maximizing that total amount of rewards notably received when interactions with the complex as well as the uncertain environment.

The overall concept of reinforcement learning has reached its peak only a couple of years ago even though artificial intelligence research has bee present in the pasty sixty years as the wide field of machine learning. Since this reinforcement learning peak, it is more than apparent that the technology industry has been greatly updating robots as well as presenting to us some very innovative machines. We really did not know that it is possible to design these machines notably able of learning by themselves with no human intervention.

This is what excites us the most, and solving some complex learning problems through reinforcement learning is simply the future. It is more than apparent that artificial intelligence techniques will shape the world that we know and reinforcement learning is a great part of different machine learning techniques as well as a great part of the artificial intelligence approach.

Reinforcement learning encompasses both adaptive behaviors in terms of rational beings and science in specific environments as well as certain computational methodology in order to find that best behavior when it comes to some challenging problems, adaptive behavior of different intelligent agents and optimization. This book welcomes you to the world of reinforcement learning by presenting you the technology lying behind self-driving cars, programs particularly able to beast world champions and being robots specifically the future and not a part of futuristic movies, not anymore.

Artificial Intelligence Python

A short Introduction to Artificial Intelligence with Python

By Anthony S. Williams

Table of Contents

Introduction

Artificial intelligence or simply AI is known as the field of machine learning. The term is referred to the intelligence notably exhibited by various machines rather than humans. The intelligence exhibited by humans is referred to as natural intelligence or NI. When it comes to the computer science, the certain field of artificial intelligence mainly defines itself as the overall study of different intelligent agents or devices specifically able to perceive their environment as well as to take different actions, which maximize the chance of success when it comes to the certain goal.

The term artificial intelligence colloquially is commonly applied in the cases when a machine or device mimics various cognitive functions which humans commonly associate with some other human minds like problem-solving and learning. The overall scope of artificial intelligence is widely disputed since machines, especially in recent times, become increasingly capable. This means that the tasks notably considered as those that require intelligence are commonly removed from the AI definition. This is known as the artificial intelligence effect widely leading to the severe grip, so we consider the AI field as everything that has not been done yet. For example, the field optical character recognition is commonly excluded from AI since it has become a certain routine technology.

When it comes to the certain capabilities of the artificial intelligence, they include competing at a high level when it comes to the strategic game systems like Go and chess, successfully understanding human speech, autonomous cars, military simulations, intelligent routing when it comes to the content delivery networks and interpreting various complex data.

Artificial intelligence, in fact, was founded as an academic discipline back in 1956. In the following years, AI has been the subject of major waves of optimism and various experiments and researches. However, in the very beginning, artificial intelligence was followed by great disappointment as well as the loss of funding specifically known as an AI winter that has been followed by some modern approaches which eventually led to renewed function.

For most of AI history, artificial intelligence field has been divided into certain subfields which often greatly fail in order to communicate with each other which is the key concept of any field. However, in the early 21st century several statistical approaches to machine learning were able to become successful enough in order to eclipse several other tools, problems, approaches and schools of thought which served as a fertile ground for further development and integration of various artificial intelligence approaches.

The traditional problems, as well as goals of artificial intelligence research, include natural language processing, reasoning, planning, learning, knowledge and the ability and perception when it comes to the moving and manipulating objects. General intelligence is the field with the long-term goals and its approaches include traditional symbolic artificial intelligence, statistical methods, and computational intelligence. When it comes to the tools commonly used in the fields of AI, they include different versions of mathematical and search optimization, methods based on statistics, neural networks, economics as well as probability.

The artificial intelligence field notably draws upon computer science, philosophy, neuroscience, psychology, linguistics, artificial psychology as well as many others. The field of artificial intelligence was founded on the various claims that human intelligence is a kind that can be described very precisely, so a machine or device is able to simulate human intelligence. This, in fact, raises various philosophical arguments when it comes to the nature of the human mind as well as various arguments about the ethics of creating some artificial beings notably with human-like intelligence.

There are many issues regarding the creation of these human-like devices and machines, which raise many issues that have been explored by philosophy, myth as well as fiction since antiquity.

There are some people that consider AI approaches and methods as a certain danger to humanity in a case if AI progresses unabatedly.

In the 21st century, various artificial intelligence techniques and methods have experienced certain resurgence mainly following concurrent advances in terms of computer power, big data, and theoretical understanding. Many AI methods and techniques recently have become a very significant as well as among the essential parts of the overall technology industry. AI techniques are those notably helping when it comes to the solving various challenging problems especially when it comes to the computer science.

Artificial Intelligence	
Research Fields	Research Techniques
Intelligent Machine	Artificial Neural Networks
Knowledge Representation	Evolutionary Computing
Machine Learning	Expert Systems
Computer Vision	Fuzzy Logic
Planning and Guidance	Genetic Algorithm
Robotics	Probabilistic Computing

When it comes to the main goals of AI, the first one I have to mention is to create technology, which allows machines and computers to function in a certain intelligent manner. However, the general problem of creating or simulating intelligence commonly has been broken down into some sub-problems notably consisting of capabilities as well as certain traits specifically expected by the researchers when it comes to the displaying an intelligent system.

The greatest emphasize in on the learning and planning which are relevant as well as easily applicable to various situations. In the following sections of the book, goals of the AI are discussed as well as various AI approaches.

Chapter 1 Approaches and Goals of Artificial Intelligence

Over the course of the last sixty years, the AI research field greatly spurred some immense features notably those which are not conceived as artificial intelligence by the general public. Majority of online endeavors include different forms of artificial intelligence like targeted advertising, pattern recognition, and various virtual agents. However, all that has been greatly done so far. Therefore, we need to acquire new knowledge on the different processes in order to position ourselves in accordance to these advancements.

When it comes to the importance of AI techniques, for instance, the business enterprise has become aware that AI methods and techniques can be and most certainly will be in the future a certain definitive factor when it comes to the overall success. These various properties currently are widely implemented in different data analysis algorithms that have a great capability when it comes to the properly storing, analyzing as well as processing Big Data that is another growing sector of business management. These recent approaches are expected to soon include different product optimization algorithms as well as complex customer engagement methods and techniques.

Origins of Artificial Intelligence

Foundations of different ideas revolving around the origins of AI can be tracked down to complex automation built by Chinese and Egyptian civilizations as well as to Greek mythology. Implementing various human properties to different objects as well as implementing abstract ideas is a certain way people have been reasoning since the very beginning of human existence as soon as they acquired consciousness.

With the emergence of the complex symbolic reasoning as well as with the development of logic, the creation of devices and machines which could emulate complex human intelligence became possible as well as achievable in practice. When it comes to the symbolic reasoning, it states that various symbols including numbers, calculations, statistics, and graphs can be easily used as certain synonymous substitutes for some longer expressions to solve different complex problems. This idea was introduced back in the 16th century by the Grandfather of artificial intelligence, Thomas Hobbes.

Further, when it comes to the origins of artificial intelligence methods since engineering advanced increasingly over the centuries, the fields of AI and engineering began to greatly correlate. The first computer was designed back in the 19th century, but it was not built until 1991. People were aware of the ongoing process when it comes to the technological advancements in the early 20th century. They also understood the increasing necessity of understanding the processes of device computing and various models which led to creating of different theoretical discourses.

When it comes to the origins of artificial intelligence, I have to mention Alan Turing who published a fundamental work on this subject back in 1950 named the Computing Machinery and Intelligence paper. In his paper, Alan proposed certain Turing machine model. In his paper, his further discussed various theoretical possibilities of different things notably what can be easily computed. He created the Turing test in order to deduct whether various computing possibilities can extend to the different spheres in terms of human intelligence.

The Turing test, proposed by Alan has an objective to test as well as to identify whether a device or machine can convince an interrogator that it was a human being. The test initially seemed to be very simple with no complex assignments like creating original art.

Further, the computer was able to perform a small talk in order to pass the test. The computer also was able to understand the given context while interacting with a human. It really sounds very simple from our perspective, but the actual realization of such results proved to be more complex and difficult than anticipated and this complexity remained up to this date. It, in fact, is unachievable.

Primary problems related to different hardware technology when it comes to the 20th century are complex storage room issues that camouflaged complex future issues in terms of software realization. Researchers are still trying to design software which would pass Alan's Turing test. The Leobner prize is still waiting for the software which would be able to pass the test and we hope to see one in the near future presented in the annual Turing Competition.

Artificial intelligence is the field of study based on different mathematical, philosophical, logical, cybernetic and information technology advancements. This field of study was born in 1956 with experts Marvin Minsky and John McCarthy which introduced this new study at a conference which took place at Dartmouth College. They became very prominent name when it comes to the wide-spanning effort in order to create intelligent devices and machines for the following fifty years.

In order to create intelligence, an expert first has to know and understand what intelligence is. On the other hand, some abstract definitions of intelligence state that intelligence is a property of human beings and some animals that are commonly manifested in learning through experience, reasoning, logic, an appliance of knowledge, creativity and other. This definition also states that it cannot be simply translated into some symbols and that it cannot produce sentient machinery.

Scientists were able to implement different approaches and techniques in order to build up complex artificial intelligence. One of the used approaches is the evolution of the complex chess-playing software. The fact is that it was much easier to achieve great efficiency through some brute force techniques notably meaning that the machine or computer is able to compute solution algorithms based on the principle of minimal cost when there is the maximum damage possible for a particular amount of all future moves. The chess-playing software, in fact, did not focus that greatly on building different sentiment, but it focused on some advanced search techniques as well as on to sustainable hardware for huge databases.

However, expert systems were developed in order to provide a greater expert assistance in various industries. By creating these proficient knowledge databases as well as by incorporating different machine learning software that enables devices and machines to make a various prediction as well as to provide consultation regarding obtained data, a scientist was able to broaden the properties when it comes to their intelligent devices and machines. Interaction software developed is based on natural language development, and all further achievements have been used in navigation systems, business management as well as in medicine to these days.

It should be noted that after the initial exhilaration within the artificial intelligence research, it soon was more than apparent that various solid results, in fact, are going to take more than previously expected and announced. Lighthill and ALPAC report clearly showed great unsatisfactory when it comes to the AI projects at the very beginning mainly due to issues with slow advancements and various natural language problems. At the time the flux of huge investments was terminated in 1974. There were no any investments to AI projects until the early 1980s when the British government decided to instigate different artificial intelligence projects in order to respond to various Japanese endeavors regarding their logic programming.

However, due to the collapses regarding the general-purpose computer market, there was a decrease in investing and funding the second Winter of AI event in 1987. In this winter periods, artificial intelligence research continued under some different names which eventually became sub-categories of the AI field including machine learning, data mining, speech recognition, industrial robotics, evolutionary programming, search engines and many other.

The main question is where is artificial intelligence these days? In order to answer this question, you just have to look around you and you will see that AI techniques and the overall artificial intelligence research enabled a great progress that is regarded common these days. The artificial intelligence research enabled specified as well as personalized search engine results, vehicle navigation systems, intelligent personal assistant software like Google Translate and Siri, diverse robotics advancements and enhancements and many other.

When it comes to the some of the most notable achievements of artificial approach and techniques, I have to mention IBM's Deep Blue which became the first computer that was able to a win a chess game against Garry Kasparov who is a chess champion. Another great achievement is IBM's complex question answerings system named Watson notably able of winning the Jeopardy quiz against some very proficient opponents back in 2011.

It is more than apparent that some hard issues of AI have not seen immense progress when it comes to the last fifty years. At the moment, many experts and scientist involved in AI projects predict at least fifty more years of error and trial in order to successfully emulate human intelligence. It is too broad as well as to complex of a subject in order to resolve in a short period of time. On the other hand, the advances which were made during the last fifty years have greatly influenced as well as shaped the world we live in. There are also many chances yet to come since the fields of AI are being explored increasingly especially when it comes to the recent years.

Artificial Intelligence Timeline:

- *Programmable mechanical calculating machine-1842*

- *Turing test by Alan Turing-1950*
- *Artificial intelligence and the first conference by Malvin Nimsky and John McCarthy-1956*
- *General problem solver demonstrated on by Newell-1957*
- *Industrial robot working on GE-1961*
- *ELIZA and the first expert system introduced by Joseph Waizenbau-1965*
- *MacHack chess-playing programm at MIT-1968*
- *Knowledge-based medical diagnosis program by Jack Myers-1979*
- *Commercial expert system-1980s*
- *Behaviour-based robotics Polly by Ian Horswill-1993*
- *Recommendation technology by TiVo Suggestions-2005*
- *Mobile recommendation apps Cortana, Siri and Now by Microsoft, Apple and Google-2011*
- *Machine learning and deep learning-present day*

The Research Goals of Artificial Intelligence

When it comes to the overall research goal of AI, it is to create an intelligent device or machine notably capable of planning, reasoning and solving various problems. The created machine should be able to learn from experience as well as to learn quickly and comprehend various complex ideas. Learning from experience and capability of planning, reasoning, thinking abstractly and learning quickly is an agreed definition of human intelligence. In practice, artificially emulated intelligence is mainly to reflect a certain broad as well as a deep ability in order to comprehend its environment and surrounding and to figure out what to do next when there are infinite possible situations.

The artificial intelligence needs to be widely socially intelligent in order to position itself in different environments. This means that AI has to be able to widely perceive as well as to properly react to some broad spectra of different abstract features as well as properties of the universe like emotion. The artificial intelligence also needs to have a capability of implementing creativity when it comes to its functioning in order to manage various problems in an optimal manner and anticipated manner. All of these needed properties are further attributed to the overall as well as to the final goal of every AI research and study.

In order to achieve any goal when it comes to the AI research, experts and scientist have to mainly focus on a huge portfolio of different and complex concepts which are notably different blocks existing individually as well as in correlation. The builders when it comes to the future intelligent machine, need to successfully implement their work into some empirical steeds of already existing intelligent systems specifically systems in term of human beings as well as we the results of different theoretical exploration and different analysis of various possible systems of intelligence including their representations and mechanism.

The factors from the previous paragraph, in fact, are the most essential when it comes to the finding a resolution of issues notably related to some already existing intelligent system. These factors are the most relevant and of great importance when it comes to the designing some entirely new intelligent as well as some semi-intelligent devices and machines. This means that the complexity of any task has to be acquired since by restricting different endeavors to just one field like engineering, the overall efforts commonly will not provide much-anticipated results. The most obvious example is that is would have been impossible to construct plane without examination of birds, and when it comes to the AI, the situation is the same, and scientists need to explore different fields of AI in order to get the anticipated results.

Goals of Artificial Intelligence:

- *Deduction, reasoning and problem solving*
- *Knowledge representation*
- *Planning*

- *Learning*
- *Natural language processing*
- *Motion and manipulation*
- *Perception*
- *Creativity*
- *Social intelligence*
- *General intelligence*

The main goals of AI have presented above. It is the right time to get to know these goals a little better in order to understand AI approach better before we step into the real world of problem-solving using AI techniques. In the beginning of artificial intelligence research, the overall reasoning process induced by step by step imitation of different human processing in terms of logical deduction and solving puzzles.

However, this certain approach greatly depended on various computation resources as well as on computer memory which ate at the time greatly confined. These issues, in fact, pointed out the great necessity existing when it comes to the imitation as well as immediate judgment process rather than a necessity of deliberate reasoning. Immediate judgment may be seen as subconscious knowledge and as well as the intuitive knowledge that governs the direction of every deliberate action.

Artificial intelligence makes different attempts at reaching its goals especially when it comes to the immediate judgment using the combination of sensorimotor skills, embodied agents, neural networks, and different statistical approaches. For instance, embodied agents are used commonly since they are greatly autonomous entities that have the ability to interact with their surroundings and environment and they are commonly presented as three-dimensional certain robot bodies.

On the other hand, sensorimotor skills are very much needed a combination of skills used when it comes to the perceiving surroundings and environment through various sensors.

Sensorimotor skills also include the ability to react with different motor skills. For instance, there is a robot notably able of perceiving an approaching person. The robot is also capable of offering a hand while greeting with the person. Neural networks represent a simulation of different structures as well as processes within the neural systems, notably human brain. Neural networks are able of computing different values obtained from inputs, while machine learning techniques are able of pattern recognition. Experts also use different statistical approaches in order to obtain specific problem resolution using mathematical approach.

Artificial intelligence has to incorporate immense amounts of knowledge regarding different objects and its properties as well as a relation between different objects. Only by doing so, AI is able to emulate a human being. Moreover, artificial intelligence has to implement different states, effects, situation and abstract ideas. The artificial intelligence research field uses a certain ontological approach in order to move towards knowledge representation which is knowledge specifically postulated in different sets of concepts whose relationship is commonly defined within a domain.

There are many issues existing when it comes to the knowledge representation like the impossibility of false or true statement since everything has exceptions. Another issue regarding this subject is the width of overall human knowledge notably making creating more comprehensive ontology almost not possible. Other issues are that sub-symbolic, as well as the subconscious forms of knowledge, needs to be integrated within AI project. Fortunately, these issues may be overcome using AI techniques like statistical AI notably using a mathematical resolution of particular problems. On the other hand, situated AI research uses various systems as autonomous entities through the certain interaction with surroundings in order to develop and obtain Elementary behaviors. Computational intelligence research is creating a computer that can understand concepts, so the computer is able to provide ontology by itself via different ways like the Internet.

It should be noted that AI has to be able to construct various complex as well as optimized solutions when it comes to the multidimensional perform realization and multidimensional space of different sequences or strategies of the overall action. It can be said in another word as well. The intelligent agents have to be able to visualize different potential failures by using predictive analysis. The intelligent agents also have to be able to visualize set goals of the overall action which belong to decision making. From the intelligent agents, it is also expected to perform in a certain manner that will greatly maximize value or efficiency of the overall process.

These goals of artificial intelligence need to be handled both online for different unexpected environments as well as offline for some already known environments. Experts still embark on some great challenges when it comes to the dealing with the issues of various unpredicted scenarios like when the machine is anticipated to react more intelligently.

Artificial intelligence is greatly correlated with the field of machine learning since machine learning is a certain construction as well as a study of different algorithms that allow artificial intelligence systems to make the different decision as well as predictions notably based on obtained data input and certain knowledge specifically acquired through machine learning process. Machine learning techniques may be focused on unsupervised and supervised methods. The unsupervised process is a pattern recognition while supervised or programmer is classification as well as a relation formation within the input data like guiding spam and non-spam emails into certain different categories within the system.

Machine learning is widely used in different spheres when it comes to the information technology. The techniques of machine learning are used in search engines personalization, optical character recognition, computer vision as well as data mining. It is expected that the further enhancements of the approaches will greatly attribute to the computational intelligence of devices and machines.

I have to mention natural language processing and machine perception since these approaches are commonly used in different AI research projects. Natural language processing, as well as generation, are one of the fundamental issues that the AI field of study commonly deals with. On the other hand machine perception widely represents the ability of different input interpretation which resembles processes of different human perception through senses.

The issues which are trying to be addressed and solved are commonly those of transmission to an intelligent core of some entity and comprehensive perception. Machine perception has to deal with different challenges in both computing features and engineering. Machine perception uses touch, vision, and hearing in order to collect different information based on audio and image properties in order to address the to various solutions or algorithms for a certain problem.

Approaches to Defining AI Systems

From the very beginning of the AI research back in the 1950s, there have been numerous approaches conducted through the implementation of different knowledge in various academic circles and industries. These approaches rapidly evolved as a certain response to some shortcomings which each of them commonly showed especially when it comes to the realization of the ultimate goal notably general intelligence. When the artificial intelligence research lost its findings during the event of AI winters, the great disintegration of different AI approaches was the only possible way in order to acquire some investments for further studies.

From today's point of view, it can be easily concluded that all of the AI approaches are relevant as well as greatly essential when it comes to the vast complexities of AI and all of them served as a great contribution to the overall process regardless of how lacking and slow advancements may be during the overall process. Connectivity approach was used in order to combine knowledge and techniques commonly used in information technology and neurology, so experts were able to achieve a simulation of the most basic intelligence back in the 1950s. Connectivity approach abounded before it has emerged again back in the 1980s. When it comes to the achievements of this approach, I have to mention that the experts were able to gain valuable knowledge on regulatory systems, sensory processing and they were able to better understand the behavior of neural networks.

Another commonly used AI approach is symbolism. This approach states that complexity of human intelligence may be simulated through manipulation of different symbols. The approach had a huge success in high intelligence simulation back in the 1960s. When it comes to the approach achievements, the scientists were able to develop complex expert systems.

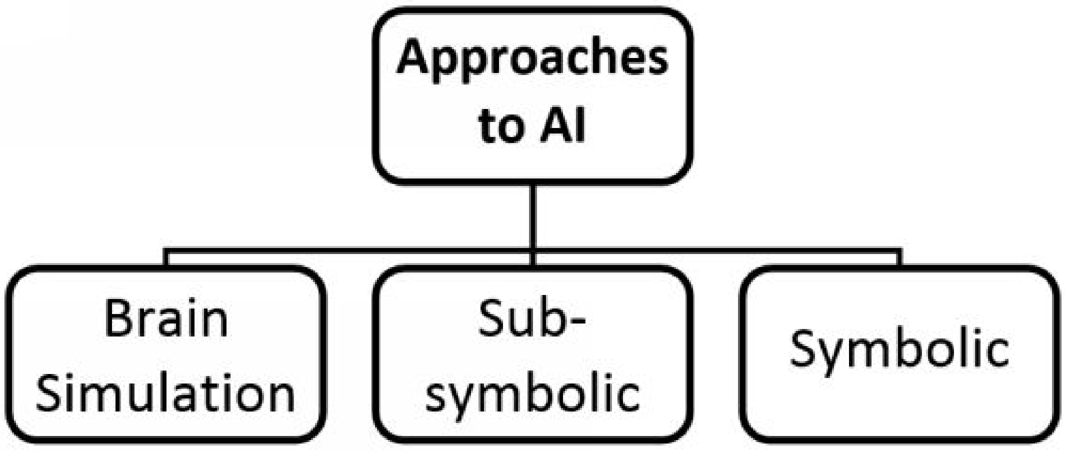
Another commonly used approach is a cognitive simulation which is embodied in different psychological tests which were commonly conducted to acquire knowledge of solving skills by a human. The results obtained were formalized in order to develop different programs notably having the ability to simulate certain properties of high human intelligence. The achievements of cognitive simulation approach include the conducting of different foundations for AI research including natural language processing and machine learning.

The logic approach is another commonly used AI approach which has that high human intelligence in its core greatly spurs from different abstract reasoning as well as from problem-solving so it can be treated with different logic's techniques. Achievements of logic approach include automated planning, logic programming, knowledge representation and machine learning.

On the other hand, there is an anti-logic approach which is the complete opposition to the logic approach. This approach states that there is no any general principle notably able of capturing the complexity when it comes to the intelligent behavior. Achievement of the anti-logic approach is that pointed out the great lack of efficiency when it comes to the logic approach especially in a matter of natural language processing and machine vision.

The knowledge-based approach is another approach that was widely implemented in various artificial intelligence studies since the increase of storage capacities and due to the emergence of different expert systems. Knowledge-based approach explored some fundamental elements of general intelligence as well as it was able to implement AI research techniques into expert systems. Another approach commonly used is the abstract approach that emerged from the great necessity of addressing fundamental specters and sub-symbolic specters of human intelligence to provide some optimal solutions for issues and problems of artificial intelligence approach. Abstract approach achievements include robotics, pattern recognition, computer perception and machine learning.

Novel or situated artificial intelligence approach mainly focuses on some basic engineering problems. The approach rejects the exclusivity when it comes to the previous symbolic approach. The ultimate goal is to construct a certain realistic machine which can exist in some real environment. Situated or novel AI approach achievements include computer perception, sensory and motor skills.The last approach I am going to mention is statistical AI approach that uses verifiable and measurable mathematical tools and further combines them with different economics tools to solve a certain AI problem. However, this approach is commonly criticized in terms of disregard towards the ultimate goal of general intelligence. On the other hand, this approach gave scientist the ability to successfully address a certain AI problems.


Approaches to AI
Brain Simulation
Sub-symbolic
Symbolic

Chapter 2 Fundamental AI Techniques

According to a certain area of AI research, there are different techniques commonly used such as natural language processing, intelligent data analysis in medicine, knowledge-base systems, cognitive modeling and other. In this section of the book, we are zooming into fundamental AI techniques in order to get to know the fundamentals of the AI methodology so most certainly you will much closer to the world of the artificial intelligence by the end of this chapter. In this chapter of the book, we will get to know AI techniques including heuristics, support vector machines, artificial neural networks, Markov decision process and natural language processing.

Heuristics

Heuristics is a technique or widely used approach when it comes to the learning, problem-solving of just discovery of which employs some practical method, but with no guarantee that the approach will result in a perfect or optimal solution. On the other hand, this AI technique provides sufficient source for some immediate goals. In the cases where finding some optimal solution is almost impossible as well as impractical, this technique is used in order to speed the overall process when it comes to the finding the most satisfactory solution.

Heuristics also may be some mental shortcut. Examples of this AI technique include using guesstimate, common sense, a rule of thumb, intuitive judgment and profiling. Scientists commonly use availability heuristic, representative heuristic, anchoring and adjustment, familiarity, escalation of commitment and naïve diversification. This technique is also found to be used when it comes to the creation and manipulation of cognitive maps. Cognitive maps are some internal representations of the human physical environment, commonly associated with different spatial relationships.

Internal representations of the human environment are widely used as memory when it comes to the guiding different human external environments. In computer science, mathematical optimization and artificial intelligence, this technique is used in order to solve problems in an easier manner and more quickly especially when using some more traditional methods are too slow. Traditional methods may fail when it comes to the finding the exact solution, so heuristic is used in order to minimize the probability of failure notably achieved by completeness, precision for speed, accuracy, and optimality.

The main objective of this technique is to create a solution in some reasonable time frame which is good enough for solving future problems. It should be noted that this solution may not be the best one, but it will be able to approximate the proper solution for a given problem. The technique is very valuable in AI research since it does not require a certain prohibitively long time. The technique is able to produce results by themselves that may be used in different conjunction with various optimization algorithms in order to improve their efficiency. This technique, in fact, underlines the whole world of AI as well as the world of the computer thinking and simulation, since this approach can be used in various situations when there are no any known algorithms.

Further, you will get to know how to create an eight-puzzle solver in Python with plug in heuristics. The model will be able to solve a randomized 8-puzzle.

```
import random

import math

_goal_state = [[ 1,2,3 ], [ 4,5,6 ], [ 7,8,0 ]]

def index (item, seq

return -1
```

This is a helper function, which returns -1 for all non-found index value of seq.

```
def __init__(self)
self._hval = 0
self. _ depth = 0
self. _ parent = None
self . adj _ matrix = []
self . adj _ matrix . append
def __ eq__
self . adj _ matrix = = other . adj _ matrix def __ str __(self): res = ” for row in range(3): res += ‘ ‘.join (map(str, self . adj _ matrix[row]))
res += ‘\r\n’ return res def _ clone (self):
p = EightPuzzle ()
p . adj _ matrix [i] = self . adj  _ matrix [i] [:]
return p
def _get _ legal _ moves (self)
row, col = self . find(0)
free = []
```

Find which pieces can move there if row is bigger than 0.

```
Free . append (( row - 1, col ))
Free . append (( row, col – 1 ))
Free . append (( row + 1, col ))
Free . append (( row, col + 1 ))
def _ generate _ moves(self)
free = self._ get _ legal _ moves ()
zero = self . find (0)
```

```
def swap _ and _ clone (a, b)
p = self . _ clone ()
p . swap (a , b)
p. _ depth = self . _depth + 1
p. _parent = self
return p
return map (lambda pair: swap _ and _ clone (zero, pair), free)
def _ generate _ solution _path (self, path)
self. _parent = = None
path . append(self)
return self. _parent. _ generate _ solution _path (path)
def solve(self, h)
def is_solved (puzzle)
return puzzle . adj _ matrix == _ goal _ state
openl = [self]
closedl = []
move _ count = 0
len(openl) > 0:
x = openl . pop (0)
move _ count += 1
len ( closed ) > 0
return x. _  generate  _  solution  _  path([])
return [ x ]
succ = x. _ generate _ moves()
idx  _ open = idx _ closed = -1
idx  _ open = index ( move, openl )
idx  _ closed = index ( move, closed ) hval = h (move) fval = hval +
move. _ depth
idx  _ open == -1:
```

```
idx  _ open > -1: copy = openl [idx _ open]
hval + copy. _ depth
hval = hval copy. _parent = move. _parent
copy. _ depth = move. _ depth
elif idx  _ closed > -1
copy = closed [ idx _ closed]
move. _ hval = hval
closedl . remove (copy)
openl . append (move)
closedl . append (x)
openl = sorted ( openl, key=lambda p: p._hval + p. _ depth)
return [], 0
```

Support Vector Machines

In artificial intelligence and machine learning, SVM's or support vector machines are common supervised learning models with different associated learning algorithms which analyze data notably used for regression and classification analysis. If you have a set of various training examples and each of them is marked as belonging to a single or to the other two categories, a support vector machine training algorithm will build a new model notably assigning some new examples to that category while making it binary linear classifier.

Every SVM model in a probabilistic classification setting is a representation of these examples commonly presented as certain points in space that are further mapped so that the different examples of these separate categories are divided by some clear gap specifically as wide as possible. Further, new examples are mapped into a certain space as well as predicted to belong to a particular category notably based on which side of the gap the examples fall. In addition to commonly performing linear classification, support vector machines may efficiently perform a non-linear classification when using what is called the kernel trick, so they are mapping a total number of their inputs into some high-dimensional feature spaces.

In the case when you have data specifically data not labeled, supervised learning will not be possible, so you have to perform an unsupervised learning that commonly attempts to find some natural clustering of the data collections, and then further map obtained data in order to find formed collections of this data. The clustering algorithm provides a greater improvement to the SVMs, so we refer to them as support vector clustering especially used in several industrial applications specifically when you have data specifically not labeled or when only some parts of the data are labeled for the further preprocessing pass.

When it comes to the implementing support vector machines in Python, we will use scikit-learns specifically widely used library when it comes to the implementing different AI projects. Fortunately, support vector machines are also available in this Python library, so you will follow the same structure like in the code presented below.

```
from sklearn import svm
model = svm . svc ( kernel=‘linear’, c=1, gamma=1 )
model . fit ( X, y )
model . score ( X, y )
model . predict ( x _ test )
```

The following step is to tune parameters of support vector machine which will effectively improve the overall model performance. If you have a linear kernel you will need the code listing from the below.

```
Sklearn . svm . SVC (C=1.0, kernel=‘rbf’, degree=3, gamma=0.0,
coef0=0.0, shrinking = True
probability = False,
tol= 0.001, cache _ size=200,
class  _  weight = None,
verbose = False,
max _ iter =-1,
random _ state  = None)
import numpy as np
import matplotlib . pyplot as plt
from sklearn import svm, datasets
iris = datasets . load _ iris ()
X = iris . data [:, :2]
y = iris . target
C = 1.0
Svm . SVC ( kernel = 'linear', C=1, gamma=0 )
.fit( X, y ) x _ min,
X _ max = X [:, 0] .min() - 1, X [:, 0] .max() + 1 y _ min, y _ max =
X [:, 1].min() - 1, X [:, 1]. max() + 1
h = (x _ max / x _ min)/ 100 xx,
yy = np . meshgrid (np . arrange ( x _ min, x _ max, h ),
np . arange (y _ min, y _ max, h)) plt . subplot (1, 1, 1)
Z = svc . predict ( np.c_[xx . ravel(), yy . ravel()]) Z =
Z . reshape (xx . shape)
Plt . contourf ( xx, yy, Z, cmap = plt . cm . Paired, alpha=0.8 )
plt . scatter ( X[:, 0], X[:, 1], c=y, cmap = plt . cm . Paired )
plt . xlabel ( 'Sepal length' )
plt . ylabel ( 'Sepal width' )
plt . xlim ( xx . min (), xx . max ())
plt . title ( 'SVC with linear kernel' )
plt . show ()
```

Artificial Neural Networks

Artificial neural networks are commonly known as connectionist systems notably computing different system widely inspired by the biological neural networks especially to those constituting animal brains. These systems are capable of learning as well as progressively improving performance in order to do different tasks by considering some examples but without task-specific programming. For instance, in image recognition, ANNs might easily learn how to identify images, which contain cats simply by analyzing different example images which have been manually labeled as cat images. ANNs further use these analytics results in order to identify cats in other obtained images. The ANNs are commonly used in different applications especially when it is complex to express using rule-based programming in a traditional computer algorithm.

Every artificial neural network is based on a certain collection of different connected units known as artificial neurons notably analogous to axons contained in a biological brain. Each synapse or connection between these neurons may easily transmit a certain signal to any another neuron.

The receiving neuron or postsynaptic further processes these signals and finally the signals downstream neurons notably connected to it. It should be noted that neurons may have their state commonly represented by real numbers between 0 and 1.

Synapses, as well as neurons, commonly have their weight, which varies as learning proceeds. The strength of the neurons can decrease or increase when it comes to the signal which sends neurons downstream. Neurons and synapses further may have a threshold like that if an only certain aggregate signal is above or below, the particular level downstream specific signal sent.

Commonly, neurons are organized in certain layers. It should be noted that the different neuron layer has a capability of performing different kinds of transformations when it comes to their inputs. Signals travel from the initial input to the last layer known as output, commonly after traversing these layers several times. The ultimate goal of the artificial neural networks technique is to solve different problems in the exact same way that a human brain does. Over the time, the approach focused on matching certain mental abilities, which led to deviations from biology like backpropagation. Artificial neural networks are widely used for different AI projects including tasks like social network filtering, speech recognition, computer vision, machine translation, playing vie and board games, medical diagnosis and many other fields.

The following line of code will you how to build a simple artificial neural network in Python.

```
from numpy import exp, array, random, dot
training _ set _ inputs = array ( [[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]] )
training _ set _ outputs = array ( [[0, 1, 1, 0]] ). T
random . seed (1) synaptic _ weights = 2 * random . random ( (3, 1) ) - 1
output = 1 / (1 + exp (- (dot( training _ set _ inputs, synaptic _ weights ))))
synaptic _ weights += dot(training _ set _ inputs . T, (training _ set _ outputs - output)
print 1 / (1 + exp (- ( dot(array ([1, 0, 0]), synaptic _ weights ))))
```

Markov Decision Process

Markov decision process or simply MDP provides a certain mathematical framework when it comes to the modeling decision making in a different situation when you have outcomes notably partly random and partly under the decision maker control. Markov decision process is commonly used when you are studying a wide range of different optimization problems commonly solved by reinforcement learning and dynamic programming.

MDPs have been used since the 1950s and the approach was introduced by Bellman in 1957 where he conducted a core body research on MDPs which eventually led to publishing a book on the subject by Ronald A. Howard in 1960. When it comes to their area of use, MDPs are commonly used in different areas of AI including robotic as well as some other areas such as economics, manufacturing, and automated control.

To be more precise, A Markov Decision Process is a certain discrete time stochastic process. It should be noted that every MDP at each step is in a certain state, so decision maker can choose any action available at MDP's state. The process further responds at the next time just by moving into some new stage as well as giving the decision maker a particular corresponding reward. It should be noted that the probability of the process transmitting into new stage depends and is influenced the certain chosen actions.

In other words, the following stage is greatly influenced and depends on the current stage. The process is an extension of complex Markov chains with the difference when it comes to the number of actions available that are allowing rewards or giving motivation as well as allowing choice. It should be noted that MDP can reduce to Markov chain is only single action exist for certain stage while all rewards available are the same.

In order to implement Markov decision process in Python depending on your problem, you may need an interface in order to determine the MDP.

```
class MarkovDecisionProcess
def transition ( self, from _ state, action, to _ state )
raise NotImplementedError def initial _ state ( self )
raise NotImplementedError def reward ( self, state )
```

raise NotImplementedError def discount (self, state)

raise NotImplementedError

You may also need the code for the modified dict class.

class SumDict (dict)

def __ setitem__ (self, key, value) self . has _ key (key)

value += self . get (key) dic . __ setitem __ (self, key, value)

After this step you are able to define the MDP.

import numpy as np

class ToyMDP (MarkovDecisionProcess)

def __init__(self)

self . world = np . array ([[-0.04, -0.04, -0.04, 1],

[-0.04, None, -0.04, -1],

[-0.04, -0.04, -0.04, -0.04],])

Self . initial _ state = (0, 0)

Self . finals = [(0,3), (1,3)]

Self . actions = ('l', 'r', 'u', 'd') def __iter__(self)

class Iterator: def __init__(self, iterator, finals)

self . iterator = iterator

self . finals = finals def next (self)

True: coords = self . iterator . coords

val = self . iterator . next()

return coords, val

return Iterator (self . world . flat, self . finals)

def _move (self, state, action)

shape = self . world . shape

```
next = list ( state )
elif action == ‘l’ and \ (state[1] > 0 and self . world[state[ 0 ]]
[state [1]-1 ] != None):
next [1] -= 1
d = SumDict ()
d [self. _ move (state, ‘l’)] = 0.8
d [self. _ move (state, ‘u’)] = 0.1 d [self . _ move ( state, ‘d’ )] =
0.1
return self . successors  ( from _ state, action )[ to _ state ]
def initial _ state (self)
return self . initial _ state def reward ( self, state )
return self . world [state [0] ] [ state[1]] def discount ( self )
return 1
```

Natural Language Processing

Natural language processing or simply NLP has commonly used techniques in different fields including computational linguistics, artificial intelligence and computer science mainly concerned with the different interactions between human or natural language and computers. Natural language processing I also concerned with programming computers in order to process huge natural language corpora. When it comes to the challenges in natural language processing, they commonly involve connecting language to machine perception, natural language understanding, machine-readable logical forms, dialog systems, natural language generation and others.

This technique is used in order to refer from speech recognition to certain language generation where every step requires a different natural language technique like Named Entity Recognition or Part-of-Speech tagging. In the following section, you will see how to build your own natural language processing assistant in Python. Before we get to know the software better, you should try natural language processing online. When it comes to the software, we will use Python's NLTK commonly used library notably containing a lot or already built in functions as well as a collection of text that will help you to get started with your NLP model. Before you embark on this adventure, make sure that you have NLTK library installed.

In order to begin, you will need to tokenize a certain sentence specifically enabling you to handle different individual words as well as punctuation marks within your sentence.

Tokenization

```
from nltk import word _ tokenize
sentence = " What is the weather in Los Angeles? "
tokens = word _ tokenize (sentence)
```

As soon as you have your tokens ready for the further processing, you can move to the next step of stop word removal. This step involves the removal all the words that are unnecessary and that simply do not contribute to any semantic meaning. Those are the words like an, the, and, etc. Python library NLTK already provides a collection of inbuilt stop words for eleven languages.

Removing the unbuilt NLTK stop words

```
from nltk . corpus import stopwords
stop _ words = set (stopwords . words(' english '))
clean _ tokens = [ w for w in tokens if not w in stop _ words ]
clean _ tokens [ 'What', 'weather', 'Los Angeles', '?' ]
```

One of the most essential parts when it comes to the speech recognition is speech tagging. In this stage, you are able to tag every word within a sentence as an adjective, noun, verb, etc. In order to do this, you will use NLTK function particularly those able to perform Parts of Speech tagging.

```
import nltk
tagged = nltk . pos _ tag (clean  _  tokens)
tagged
[ ( 'What', 'WP' ), ( ' weather ', 'NN' ), ( ' Los Angeles ', 'NNP' ), ('?', '.')]
```

This tagged list contains different tuples of the form or word. When it comes to the next step, you will need NER or named entity recognition, which also goes by names Entity Extraction or Entity Identification. We need this step since Named Entity Recognition involves identifying every named entity.

NER further puts all named entities into different categories like the name of an organization, a location, etc. In order to perform NER, you will use NLTK function that classifies all named entities. In the following function, Parts of Speech tagging are represented as the input of the function.

```
Print ( nltk . ne _ chunk( tagged ))
( S What/WP weather/NN ( GPE Los Angeles/NNP ) ?/.)
```

After you named entity recognition, the overall meaning of the sentence is entirely analyzed, so you will make the appropriate call to you API. For example, in the sentence used here, after recognizing the location as Los Angeles, the context of weather may be made to some cloud based service. Then, the current weather will be displayed to the user who asked the question about the weather in Los Angeles. This is the code listing in order to get started with natural language processing using NLTK Python library. This technology is used in major intelligent personal assistants such as Siri, Amazon Alexa, and Google Now.

Chapter 3 Artificial Intelligence Methodology

Reinforcement Learning

Reinforcement learning is one of the most important questions for experts and scientist who are involved in artificial intelligence problems and projects. It is more than apparent why they desire to know these answers regarding the reinforcement learning. In they can understand these problems, they will, in fact, enable human to do things we were not able to do before. Therefore, with reinforcement learning, we will be able to train devices and machines to do more human tasks as well as to create something very close to the true artificial intelligence.

However, there is not yet a complete answer to the important question from the above, but there are things which proven to be more clear especially when it comes to the recent times. We all learn by interacting with our environment no matter what we are trying to accomplish. Whether you are learning to drive a car or whether an infant is learning to walk, in both scenarios a human will learn by noticing environment and by interacting with the environment. The foundation all many theories of learning, as well as the foundation of intelligence, are learning from the environment, and this concept is also integrated into many AI practices and researchers.

In this section of the book, we will explore reinforcement learning particularly a goal-oriented as well as based on interaction with our environment. It is said that reinforcement learning is the ultimate hope when it comes to the true artificial intelligence. I have to admit it is completely rightly said so since there is a huge potential of reinforcement learning when it comes to the limitless possibilities of machine learning. This approach is growing rapidly since it produces a wide range of different learning algorithm notably used for various applications. Therefore, it is important to get to know basic concepts of reinforcement learning if you are thinking about doing some AI research or project.

Once you are there and once you have the basic knowledge of reinforcement learning, you will understand as well as be able to implement it on your own AI project. The approach is learning what to do as well as how to map different situations to actions. The ultimate goal is to maximize that numerical reward signal. It should be noted that the learner is not told which certain action to take, but he or she must discover which action may yield that maximum reward. In order to devote to reinforcement learning, you will first observe and notice things from your environment, and then you will have your first attempt when it comes to the rewards. You have to remember that with reinforcement learning, there are many things you should keep in mind like balancing different tasks, paying attention to your environment and observing and noticing things which you will use in your AI project.

It should be noted that reinforcement learning, in fact, belongs to some bigger class of machine learning algorithms. You are already aware of the fact that machine learning involves supervised, unsupervised and reinforcement learning. Supervised is task driven including regression and classification while unsupervised in data drive or clustering. On the other hand, reinforcement learning involves algorithms, which learn to react to an environment, so you notice the difference.

In order to understand how to solve some reinforcement learning problem, you have to understand and get to know the problem of exploitation and exploration and then you are ready to embark on solving your reinforcement learning problem. In order to understand this, we should suppose that we have many slot machines which offer random payouts. We need to discover and get the maximum bonus from a slot machine as quickly as possible. A naïve approach is to select just one slot machine and keep playing on it all day long. Well, it sounds more than boring, even though there is a possibility of hitting the jackpot. This approach is commonly defined as a pure exploitation, and it is not the optimal choice.

On the other hand, there is another approach, for instance, when you play on every slot machine. This approach is also not that optimal, even there is still a possibility of hitting a jackpot. This approach is defined as pure exploration. It is more than apparent that neither of this approaches is optimal since there is no proper balance in order to get that maximum rewards. This is very common exploitation vs exploration dilemma when it comes to the reinforcement learning.

In order to get close to the reward, you have to formally define framework notably for defying a solution in any reinforcement learning scenario. It can be designed as set of states noted as S, set of actions noted as A, reward function noted as R and value V. In order to properly implement reinforcement learning you will use a Deep Q-learning algorithm notably commonly used policy mainly based on learning algorithm with some function approximator as an artificial neural network. Google commonly uses this particular algorithm in order to beat humans and its Atari games. You will initialize Q as well as choose an action. You further perform an action, measure reward and update Q.

Now, it is time to integrate that into Python and code it up. You will run the following commands.

```
git clone https: //github.com/matthiasplappert/ keras-rl.git cd
keras-rl python setup.py install
```

You will install different dependencies for your environment.

```
pip install h5py
pip install gym
```

Further, you will import modules, which are necessary.

```
import numpy as np
import gym from keras . models
import Sequential
from keras . layers import Dense, Activation, Flatten
from keras . optimizers import Adam from rl . agents.dqn import
DQNAgent
from rl.policy import EpsGreedyQPolicy
from rl . memory import SequentialMemory
```

Then you will set all relevant variables.

```
ENV_NAME = ‘CartPole-v0’
env = gym . make( ENV_NAME )
np . random . seed( 123 )
env . seed( 123 )
nb _ actions = env . action _ space .n
```

You will build hidden neural network layer

```
model = Sequential()
model . add (Flatten( input_shape =(1,) + env . observation _
space . shape ))
model . add ( Dense( 16 ))
model . add (Activation( ‘relu’ ))
model . add(Dense( nb _ actions ))
model . add( Activation( ‘linear’ )) p
rint ( model . summary ())
```

Further, you will configure as well as compile your agent.

```
policy = EpsGreedyQPolicy  ()
memory = SequentialMemory ( limit=50000, window _ length=1 )
dqn = DQNAgent (model=model, nb _ actions=nb _ actions,
memory=memory, nb _ steps _ warmup =10
target _ model _ update=1e-2, policy=policy)
dqn . compile (Adam( lr=1e-3), metrics=[ ‘mae’ ])
dqn . fit( env, nb _ steps=5000, visualize = True, verbose=2)
```

Test your reinforcement learning model. In this section, you have seen a fundamental implementation of reinforcement learning performed in Python. As soon as you figure this process, you are ready to embark on other and more challenging problems.

```
Dqn . test ( env, nb _ episodes=5, visualize=True )
```

Regression and Classification

In this section of the book, we will explore classification and regression which are supervised learning problems. Since you are already familiar with these terms, we can jump immediately into Python coding and create classification and regression in Python. The problem solved in supervised learning like regression and classification will show you the link between the observed data and some external data which you will be trying to predict commonly called labels or target. In this section, we will use supervised estimators built in Python library scikit. These created prediction trees specialize in predicting outcomes. These models commonly work with estimation. In order to start, you will load your data.

```
from sklearn . datasets import load _ iris
iris = load _ iris ()
```

```
X, y = iris . data, iris . target
features = iris . feature _ names
```

As soon as you load your data into value X that contains a certain predictor and holds the classification, you will be able to define any cross-validation in order to check the results while using decision trees.

```
from sklearn . cross _ validation import cross _ val _ score
from sklearn . cross _ validation import KFold
crossvalidation = KFold (n=X.shape [0], n _folds=5
shuffle= True, random _ state=1)
```

If you use the DecisionTreeClassiffier, you will be able to define a certain max-depth specifically inside an iterative loop in order to experiment. With various effects when it comes to the increasing the complexity of the final tree. The common expectation is to reach that ideal point as soon as possible and then to witness that decreasing cross-validation overall performance due to over-fitting. It should be noted that the best solution is a tree containing four splits. As soon as you are done, you can check the complexity of your resulting tree.

```
from sklearn import
tree _ classifier = tree . DecisionTreeClassifier
max _ depth=depth, random _ state=0 )
.fit (X,y). tree _. max _ depth
score = np . mean ( cross _ val _ score (tree _ classifier, X, y,
scoring='accuracy', cv=crossvalidation, n _jobs=1 ))
print 'Depth: %i Accuracy: %.3f' % ( depth , score )
Depth: 1 Accuracy: 0.580
Depth: 2 Accuracy: 0.913
Depth: 3 Accuracy: 0.920
```

```
Depth: 4 Accuracy: 0.940
Depth: 5 Accuracy: 0.920
```

Clustering Algorithms

Clustering data is unlabeled data which may easily be a performer with the Python sklearn cluster. It should be noted that every clustering algorithm commonly comes in two different variants including a class that implements the fit function in order to learn all the clusters contained in train data, and a certain function, which returns an array of different integer labels notably corresponding to the various clusters. You will find the labels over the certain training data in the labels' attitude. In the following section, you will see K-means clustering algorithm, which will yield using three certain clusters. You will also see the effect of any bad initialization when it comes to the classification problem.

```
import numpy as np
import matplotlib . pyplot as plt
from sklearn . cluster import KMeans
from sklearn import datasets
np . random . seed(5) iris = datasets . load _ iris ()
X = iris . data
y = iris . target
estimators = [( 'k_means_iris_8', KMeans ( n _ clusters=8 ))
( 'k_means_iris_3', KMeans (n _ clusters=3 ))
( 'k _ means _ iris _ bad _ init', KMeans (n _ clusters=3, n _
init=1, init='random'))]
fignum = 1
titles = [ '8 clusters', '3 clusters', '3 clusters, bad initialization' ]
fig = plt . figure( fignum, figsize = ( 4, 3 ))
ax = Axes3D(fig, rect= [0, 0, .95, 1] , elev = 48, azim=134)
est . fit (X) labels = est . labels_
```

```
ax . scatter (X[:, 3], X[:, 0], X[:, 2], c=labels . astype (np.float),
edgecolor = 'k')

ax . w _xaxis . set _ ticklabels ([])

ax . w _yaxis . set _ ticklabels ([])

ax . w _zaxis . set _ ticklabels ([])

ax . set _xlabel ('Petal width')

ax . set _ylabel ('Sepal length')

ax . set _zlabel ('Petal length')

ax . set _ title (titles[fignum - 1])

ax . dist = 12 fignum = fignum + 1

plt . figure (fignum, figsize = ( 4, 3 ))

ax = Axes3D (fig, rect = [0, 0, .95, 1], elev=48

azim = 134) for name, label in [( 'Setosa' , 0), ('Versicolour' , 1), (
'Virginica' , 2)]

ax.text3D(X[y == label, 3]. Mean (), X[y = = label, 0]. Mean (),
X[y = = label, 2].mean() + 2, name

horizontalalignment = 'center', bbox = dict ( alpha=.2, edgecolor
='w', facecolor='w' ))

np . choose (y, [1, 2, 0]) . astype(np.float)

ax . scatter ( X[:, 3], X[:, 0], X[:, 2 ], c=y, edgecolor= 'k' )

ax . w _xaxis . set _ ticklabels  ([])

ax . w _yaxis . set _ ticklabels ([])

ax . w _zaxis . set _ ticklabels ([])

ax . set _xlabel ('Petal width')

ax . set _ylabel ('Sepal length')

ax . set_zlabel('Petal length')

ax . set _ title ('Ground Truth')

ax . dist = 12

fig . show ()
```

Chapter 4 Build a Recommender Systems

In this chapter, we will see how to build a recommender system using Python. It should be noted at the very beginning that recommendation systems are just an automated form of any other system. These systems are commonly used in a form of a shop counter guy when well-trained system sell you products in cross selling.

These systems have the ability to recommend some personalized content notably based on past behavior. It, in fact, brings customer delight as well as gives them a reason in order to keep returning to the product or brand. You can ask the system which product you may buy, and it will show you recommended products for you based on your pars behavior and past purchases.

In the following section, you will see how can you create your own recommender system using Python and GraphLab. Before embarking on this adventure, you should know that there are different types of recommender systems like those where there is no any personalization involved or those systems which use a classifier in order to make a certain recommendation.

The system which uses a classifier incorporates personalization and it works even in the cases when the user's history of purchases is short or even not available at all.

The third type or recommender system is known as recommendation algorithm especially tailor-made in order to solve different recommendation system problems including two most typical kinds, Collaborative Filtering and Content Based. In this section, we will use the MovieLens for building our recommendation system.

You will need to download dataset in order to begin. It consists of 100,000 rating from more than thousand users on more than two thousand movies. In order to start, you will load downloaded data into Python.

```
import pandas as pd
u_cols = [ ‘user _ id’, ‘age’, ‘sex’, ‘occupation’, ‘zip_code’ ]
users = pd . read _ csv(‘ml-100k/u.user’, sep=’|’, names=u _ cols,
encoding= ‘latin-1’)
r_cols = [ ‘user_id’, ‘movie_id’, ‘rating’, ‘unix_timestamp’ ]
ratings = pd . read _ csv ( ‘ml-100k/u.data’, sep=’\t’, names=r _
cols
encoding= ‘latin-1’ )
i_cols = [ ‘movie id’, ‘movie title’ ,‘release date’,‘video release
date’, ‘IMDb URL’, ‘unknown’, ‘Action’, ‘Adventure’, ‘Animation’,
‘Children's’, ‘Comedy’, ‘Crime’, ‘Documentary’, ‘Drama’,
‘Fantasy’, ‘Film-Noir’, ‘Horror’, ‘Musical’, ‘Mystery’, ‘Romance’,
‘Sci-Fi’, ‘Thriller’, ‘War’, ‘Western’]
items = pd . read _ csv (‘ml-100k/u.item’, sep=’|’
names=i _ cols
encoding= ‘latin-1’)
```

You will notice that there are exactly 943 users and you have five features for their unique ID, gender, occupation as well as the zip code of their location. You will see movies rating as well.

You will also notice that there are 100K movie ratings for various users including different movies combinations. You will also notice a certain timestamp particularly associated with every user.

```
print ratings . shape ratings.head ()
print items .  shape items . head ()
```

This certain data contains 1682 movies and there are twenty-four columns with specific movie genre. You will divide the rating into train and test data for further models.

```
R _ cols = [ ‘user_id’, ‘movie_id’, ‘rating’, ‘unix_timestamp’ ]
ratings _ base = pd . read _ csv (‘ml-100k/ua.base’, sep=’\t’
names = r _ cols, encoding = ‘latin-1’ ) ratings _ test = pd .
read_csv ( ‘ml-100k/ua.test’, sep=’\t’, names=r_cols,
encoding=‘latin-1’ )
ratings _ base . shape
ratings _ test . shape
Output: (( 90570, 4 ), ( 9430, 4 ))
```

As soon as you done, you will be able to use this obtained data for both testing and training. You will be able to make any content based system as well as collaborative filtering algorithm using this model.

Automatic Speech Recognition

In this section of the book, you will get to know better an excellent Google Speech Recognition API. This certain API converts text into written text. You will be able to simply speak and Google API will easily convert it into text and you will get excellent results written in English. Google also has created JavaScript, so you will be able to recognize your speech in Java if you like.

In order to start, you have to install packages including PyAudio, speech recognition, and PortAudio. It should be noted that PyAudio version 0.2.9. is also required and there is a strong possibility that you will have to complete the survey manually.

```
git clone
```

```
cd pyaudio sudo python setup . py install
sudo apt - get installl libportaudio - dev
sudo apt - get install python-dev
sudo apt - get install libportaudio0 libportaudio2 libportaudiocpp0
portaudio19 -dev
sudo pip3 install SpeechRecognition
```

As soon as you install the program, it will record audio coming from your microphone and it will send the certain speech API as well as return a Python string. The audio is recorded via the speech recognition module and the module is also included on the top of the program. Further, you will send your speech to the API which will return the output.

```
import speech _ recognition as sr
r = sr. Recognizer () with sr . Microphone () as source
print (“Say!”)
audio = r . listen (source)
print ( “You said: ” + r.recognize _ google ( audio ))
print ( “Google Speech Recognition could not understand audio” )
print ( “Could not request results from Google Speech Recognition
service; {0}”. format ( e ))
```

Chapter 5 Genetic and Logic Programming

Genetic and logic programming are commonly used techniques when it comes to the artificial intelligence tasks. When it comes to the genetic programming, the technique is used as a collection of genes which are modified or evolved using a certain evolutionary algorithm known as GA. It is an application of different algorithms in which space of solutions is composed of various computer programs.

The results of this approach are complex computer programs which have the capability of performing well in almost every predefined task. The method is commonly used when it comes to the encoding a computer program in any artificial intelligence chromosome. This method also leads to an evaluation of its fitness in addition to possessing a certain respect towards that predefined task which is the main concept in the GP technique. It should be noted that this technique is still a very active subject of research.

On the other hand, logic programming is a certain type of programming paradigm specifically mainly based on certain formal logic. It should be noted that any program notably written in a logic programming language is a collection of a certain sentence in logical form. Sentences written in a logical programming manner also express rules as well as fact about some problem domain. Some of the major logic programming language families include Datalog, Prolog and Answer set programming or ASP. It should be noted that in this language, all rules are written in the certain form of clauses that are read declaratively like logical implications.

When it comes to the logical programming, rules, as well as fact, have no body, so they are written in the simplified form. The simplest form is known as atomic formulae where all clauses are called Horn clauses or definite clauses. However, there are many extensions when it comes to these simplified clauses. Logic programming languages include extensions, so they also that knowledge needed when it comes to the exploring the capabilities of a non-monotonic logic. In the following section, we will see how to create logic and genetic algorithms using Python. In the followings section, genetic programming meets Python so you will how to easily fight syntax trees by implementing genetic programming in Python. The following codes are implemented in pure Python. The genetic programming code is very flexible since it compiles different GP trees in built-in Python bytecodes so the function has the great speed when it comes to the execution. You will be able to use different Python objects like terminals as well as any Python expression. It should be noted that any available Python function may be used as well. You will use Python functions that will be automatically detected by the certain framework.

```
from pyevolve import

import math

error _ accum = Util . ErrorAccumulator ()

def gp _ add (a, b) : return a+b

def gp _ sub(a, b): return a-b def gp _ mul (a, b): return a*b

def gp _ sqrt(a): return math . sqrt ( abs ( a )) def eval _func
(chromosome)

global error _ accum error _ accum . reset ()

code _ comp = chromosome . getCompiledCode () for a in xrange
(0, 5)

xrange (0, 5)

eval (code _ comp)

target = math . sqrt (( a*a )+( b*b ))

error _ accum += ( target, evaluated )

return error _ accum . getRMSE () def main _ run (): genome =
GTree.GTreeGP()

genome . setParams(max _ depth=5, method= "ramped" )

genome . evaluator.set( eval _func )

ga = GsimpleGA . GsimpleGA (genome)
```

ga . setParams(gp _ terminals = [‘a’, ‘b’], gp_function_prefix = “gp”)

ga . setGenerations (1000) ga . setMutationRate(0.08) ga.setCrossoverRate(1.0)

ga . setPopulationSize (2000) ga .evolve(freq_stats=5) print ga.bestIndividual()

As you can notice in the source-code, you will not need to bind different variable when you are calling the syntax tree of a different individual. You will simply use the code get compiled code approach that returns the Python certain compiled function to that stage where it is ready to be executed. It should be noted that the visualization is very flexible as well, especially if you use Python decorators in order to see how different function may be graphically represented. You may have different as well as very interesting visualization patterns thanks to changing the function gp_add.

Gtree . gpdec (representation=”+”, color=“red”)

def gp _ add (a, b): return a+b

Now it is the time to see how to implement logic programming approach using Python for solving a certain problem. It should be noted that logic programming is a specific general programming paradigm. Logic programming paradigm commonly came about very specifically in order to serve as widely used algorithmic core when it comes to the Computer Algebra Systems in Python. Logic paradigm also came commonly for the automated optimization and generation of numerous numeric software. The hot topics when it comes to the Python Scientific community are most certainly domain specific languages, compilers and code generation. In the following section, we will use LogPy that aims to be a low-kevel core for various projects including logic programming in AI research.

We will use LogPy which enables the expression of different relation as wekll as the search for different values that satisfy them. It should be noted that multiple variables as well as multiple goals may be used at the same tim.

```
from logpy import run, eq, membero, var, conde
x = var ()
run (1, x, eq(x, 5)) (5,)
z = var ()
run (1, x, eq(x, z), eq (z, 3)) (3,)
run (1, x, eq ((1, 2), (1, x))) (2,)
run (2, x, membero (x, (1, 2, 3)
```

It should be noted that LogPy stores different data like facts which state different relationships existing between different terms. The code form the below creates a certain parent relationship as well as uses it to state different facts about Simpsons family parents.

```
from logpy import Relation, facts
parent = Relation ()
facts (parent, ( “Homer”, “Bart” ), ... ( “Homer”, “Lisa “), ... (
“Abe”, “Homer” ))
run (1, x, parent ( x, “Bart”)) ( ‘Homer’,)
run (2, x, parent (“Homer”, x)) ( ‘Lisa’, ‘Bart’)
```

You are also able to use different intermediate variables as well when it comes to some more complex queries. You also can express the relationship with the grandfather separately.

```
y = var ()
run ( 1, x, parent (x, y), parent (y, ‘Bart’)) (‘Abe’)
```

```
def grandparent (x, z): ... y = var() ... return conde((parent (x, y),
parent (y, z)))
run ( 1, x, grandparent( x, ‘Bart’ )) ( ‘Abe,’ )
```

It should be noted that LogPy mainly depends on tuples, functions, generators, and dicts. It should be very simple when it comes to the integrating into some preexisting code since there are no new data structures or classes in LogPy.

Conclusion

We have come to the end of the book. I am sure that by now, you are familiar with different artificial intelligence approaches as well as different techniques used when it comes to the solving common as well as more complex AI problems. The book will help you when you are ready to embark on the adventure called AI and it will most certainly help you to better understand what is the machine intelligence. AI is intelligence commonly exhibited by devices and machines rather than humans. The field of artificial intelligence is defined as the study of different intelligent agents, and today it would be almost impossible to imagine the world without AI approach and techniques.

By now you now that intelligent agents are any device or machine with the capability of perceiving its environment as well as taking certain actions when it comes to the maximizing its chance of success as some certain goal. Artificial intelligence is also applied when a device or machine mimics different cognitive functions of humans. This concept is what artificial intelligence is all about. When it comes to the different applications of AI methods, I have to mention health care, video games, finance, automotive industry and much more. Try to imagine these different fields without AI approach. Yes, it is hard since we are used to AI to be present in our environment, but we simply do not pay attention to the great significance of this relatively young field of machine learning.

It is not easy being human in this age that is completely the age of artificial intelligence, but we have to accept it as well as to take advantage of it since we can greatly benefit and enhance our lifestyle. Machine learning, as well as artificial intelligence, have become mainstream tools, especially in recent times. These techniques are being applied across numerous industries in order t0 increase profits, save lives reduce costs and improve the overall customer experience. It is important to understand and get to know different AI tools and techniques, since you may benefit greatly whether you have your own business, or you simply want to explore AI by yourself in your free time.

Text Analytics with Python

A Brief Introduction to Text Analytics with Python

By Anthony S. Williams

Table of Contents

Introduction

Text analytics or text mining is all about deriving some high-quality structured data from obtained unstructured data. A very good reason for using text analytics might be extracting some additional data about customers from obtained unstructured data sources, in order to enrich that customer master data and to produce entirely new customer insight as well as to determine sentiment about different services and products. The most common text analytics applications are in case of management, for instance, healthcare patient record and insurance claims assessment. Text analytics is widely used as well in competitor analysis, media coverage analysis, sentiment analytics, voice of the customer, pharmaceutical drug trial improvement, fault management and different fields of service optimization.

Text mining or text analytics is a certain process of converting obtained unstructured data into some meaningful data notably used for further analysis especially when it comes to the measuring customer opinions, feedback and product reviews in order to provide that search facility, entity modeling, and sentimental analysis in order to support mainly fact-based decision making. Text mining uses techniques from several different fields including machine learning techniques, statistical and linguistic techniques. Text mining also involves different information particularly retrieved from unstructured data as well as the process of properly interpreting that output data in order to derive trends and patterns as well as to evaluate and interpret the output data.

Text mining commonly involves categorization, lexical analysis, pattern recognition, annotation, clustering, tagging, visualization, information extraction, predictive analytics and association analysis. This certain field of data analytics provides different tools, algorithm-based applications, extraction tools and servers used for converting unstructured data into some meaningful data notably used for further analysis. The outputs of such data analytics are some extracted entities and facts particularly with certain relationships stored in some relational XML or some other warehousing applications by various tools like big data analytics, predictive analytics or business intelligence tools.

Text mining is commonly referred to as text mining notably equivalent to text analytics. As I already mentioned it is the process of deriving some high-quality as well as relevant and useful information from various textual data. This high-quality information is commonly derived through a certain process of devising various patterns and different trends through various means like statistical pattern learning. Text analytics commonly involves the certain process of structuring different input data like parsing in addition to some already derived linguistic features as well as the removal of some other features. Further text analytics involves certain driving patterns within that structured data and eventually evaluation as well as interpretation of the certain output data.

That high-quality feature in text analytics refers to a certain combination of novelty or relevance. Common text analytics task includes text clustering, text categorization, entity extraction, document summarization, production of different granular taxonomies, entity relation modeling and sentiment analysis. This process also commonly involves lexical analysis and information retrieval when it comes to the studying word distribution and frequency, annotation and tagging, information extraction. When it comes to the overall goal of text analytics process, it is to turn some text into structured data prepared for analysis via different applications, mainly natural language processing, and different analytical techniques.

A typical text analytics application is to scan a certain collection of documents specifically written in some natural language. The following step is to model that scanned document collection for further predictive classification purposes as well as to populate that database or search index so it contains all of the information previously extracted.

Text Analytics is a certain field of data analytics commonly used in scientific papers, different business intelligence spheres and many other scientific fields regarding big data analytics. Once the text has been obtained, text analytics software engaged a certain number of different text analytics system, which aims to answer three broad questions. So, using text analytics software you can find who is talking and what are they discussing and saying and how they feel. This is very useful when it comes to the customer feedback, product, and service reviews.

Text Analytics :

- Sentiment Analysis
- Summarization
- Visualization
- Text Identification
- Text Clustering
- Search Access
- Entity or Relation Modeling
- Link Analysis
- Text Mining
- Text Categorization

Commonly, the term text analytics is used in order to describe a collection of different statistical, linguistic and machine learning techniques, which structure and model the information or different textual source for exploratory data analysis, investigation, research and business intelligence. The term is very similar to text mining. However, the term text mining is mainly used in different business settings. This term also commonly describes which application or different text analytics applications respond to certain business problems both independently and in conjunction with some query and numerical data.

It should be noted that eighty percent of various business-relevant information, in fact, originates from some unstructured data, mainly text. These text analytics techniques, as well as process, are used in order to discover and later present knowledge including business rules, facts and relationships notably locked in that textual from specifically impenetrable without text analytics to automated processing. We are witnesses to increasing interest in text analytics especially in recent times since multilingual data mining techniques can easily gain information across different languages as well as cluster some similar items from various linguistic sources in accordance with their meaning. Therefore, there is no surprise in the fact that text analytics has grown to be among the most important big data analytics techniques.

Chapter 1 Text Analytics Process

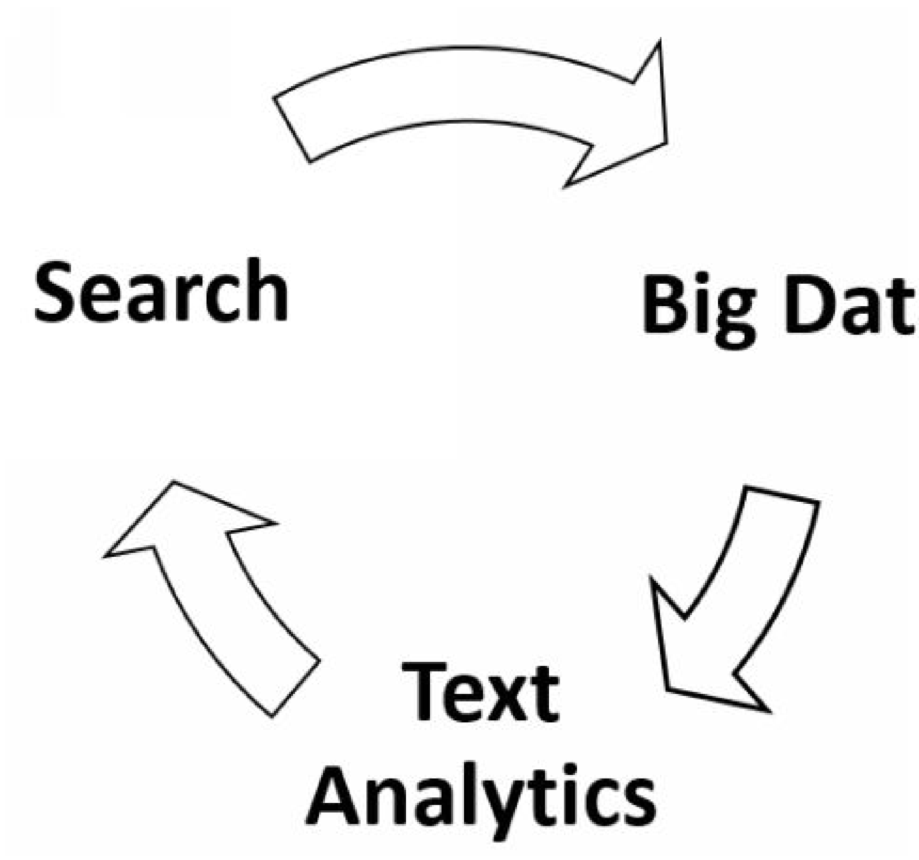
When it comes to the great challenges of exploiting that large proportion of enterprise information, which originates mainly from some unstructured form it has been recognized for decades now. These challenges are recognized in the business intelligence earliest definition back in 1958 in a journal article written by H.P. Luhn.

In his article, Luhn described a business intelligence system as a system used for utilizing different data-processing machines for auto-encoding and auto-abstracting of different documents in order to create some interesting profiles for every action point within a single organization. He also added that both internally and incoming generated documents are further automatically characterized and abstracted by a certain word pattern notably sent automatically to some proper action points.

Certain management information systems have been developed in the 1960s. At this time, IBM has emerged in the 90s as a particular software category as well as a certain field of practice notably emphasizing the importance of numerical data commonly stored in some relational databases. This was not surprising at all since text within some unstructured documents is a very challenging task to be processed easily.

The emergence of different text analytics techniques in this current form greatly stems from refocusing data analytics research, which took place back in the 1990s from algorithm development to different applications.

The fact is that for almost a decade the certain computational linguistics community simply has viewed amazingly large text collection as a certain resource notably used for tapping to produce some better quality and more useful text analytics algorithms. Later, the new emphasis on the text analysis has been introduced especially when it comes to the usage of different large online text data collection in order to discover some new trends and facts about the world we live in and about our future.



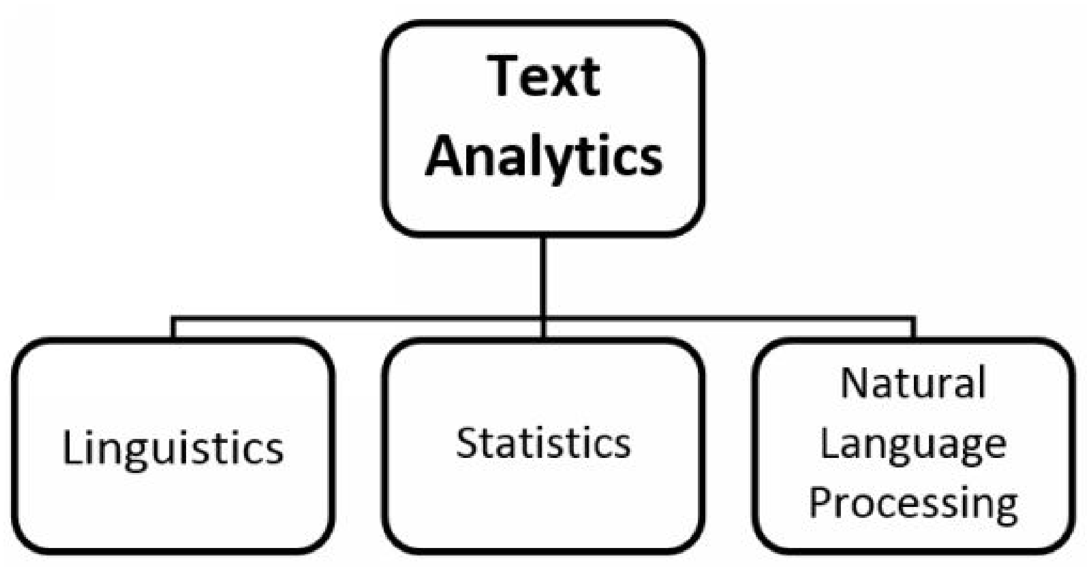
Text analytics task starts by generating different semantics in order to bridge big data and search. Text analytics applications enable different next-generation information systems. Semantic search includes obtained textual data while being mainly based on different text analytics applications. On the other hand, big data provides access to valuable and relevant information used for text analytics.

Subtasks or common text analytics process include information retrieval, natural language processing, named entity recognition, identification of certain noun phrases, sentiment analysis, quantitative text analysis and relationship, event or fact. The process of information retrieval is commonly referred to as identification of obtained corpus as an initial or preparatory step in the process of text analytics. The process includes collecting and identifying a certain collection of different textual materials commonly held in some file system or on the Web database.

There are some text analytics systems notably applying some exclusively advanced statistical methods. However, the most common methods applied in the text analytics process is natural language processing like syntactic parsing, speech tagging and other types of different linguistic analytics methods. Text analytics process commonly involves named entity recognition specifically using gazetteers and different statistical techniques in order to identify different named text features like place names, organizations, stock ticker symbols, people and various abbreviations. Disambiguation is also commonly involved in text analytics process including the use of different contextual clues notably required in order to make certain decisions.

The following step involved in text analytics process is recognition of different pattern identified entities. For instance, different features like telephone numbers, various quantities with units or e-mail addresses may be discerned using pattern matches and some other regular expressions. The next step commonly involved is coreference. It is the process of identifying different noun phases as well as other terms, which commonly refer to the exact same object. Event, relationship or fact extraction is the process of identifying different associations existing among different entities. Relationship extraction is also used when it comes to the identifying other information contained in obtained textual data.

Sentiment analysis is mandatory text analytics task notably commonly involving discerning subjective material as well as extracting different forms of some attitudinal information like emotion, mood, opinion, and sentiment. Text analytics techniques are very helpful when it comes to the analyzing various sentiments within an entity and distinguishing various opinion holders. Another commonly used task in text analytics techniques is quantitative text analysis notably representing a certain collection of different techniques stemming from some social sciences. Quantitative text analysis can give us answers to questions regarding a human judge involved in different computer extracts semantic. Quantitative text analysis also is very helpful when it comes to finding grammatical relationships between various words to find the stylistic patterns and meaning of commonly some casual personal text used for psychological profiling.



Text analytics techniques are widely applied to different spheres of business and research needs including scientific discovery including life sciences, enterprise business intelligence, competitive intelligence and data mining, e-discovery, national security, records management, sentiment analysis tools or listening platforms, publishing, social media monitoring, automated and placement, information search and information access.

Many text analytics software packages are commonly marketed for different security applications especially when it comes to the monitoring as well as analyzing some online text source like Internet news and other mainly for national security purposes. Text analytics techniques are also commonly involved in the study of text decryption and encryption.

A wide range of text analytics applications are used in biomedical text mining. On the other hand, text analytics techniques and software are commonly used as well as developed by some major companies like Microsoft and IBM in order to further automate the text analysis process. These different companies working in this field greatly contribute to improving the overall text analytics results. It is more than apparent that public sector is greatly concentrated on creating certain software notably used for monitoring and tracking some against the law activities, so text analytics has been of great help.

Text analytics methods have a wide range of online media applications used by some major companies like Tribune Company. In this specific scientific field, text analytics methods are used in order to provide readers with better search experiences that commonly increase site revenue and interestingness. Major media houses are focused on assembling and organizing their content while using faster and better semantic publishing provided by common text analytics methods. On the other hand, major editors are commonly benefiting by being capable of sharing, associating and packaging news across various properties notably greatly increasing their opportunities when it comes to the monetizing their content.

Majority of text analytics techniques and methods is starting to be used in business and marketing applications especially when it comes to the customer relationship management. Major business and marketing companies apply text analytics techniques in order to attract customers and keep their attention. Text analytics methods are also commonly used when it comes to the stock return prediction.

When it comes to the sentiment analysis, its techniques and methods are commonly used in an analysis of various movie reviews in order to estimate how favorable a certain review is for a specific movie. This kind of text analysis commonly requires some labeled data collection as well as affectivity of different words. When it comes to the certain resources for affectivity of words, analytics commonly use ConceptNet and WordNet software. Text commonly has been used in order to detect various emotions in some related are of commonly used affective computing. Text-based techniques and approaches to various affective computing methods have been used when it comes to the news stories, children stories, and student evaluations.

We are witnesses of huge amounts of text contained on Web and other file containers commonly used by big data analytics. That certain text online is creating vast relevant and useful data about many things like what is happening in the world or what different people think. All of this relevant data is an absolutely invaluable resource which can be mined in order to generate very useful business insights for organizations and analysts. However, in order to analyze all of this obtained content is very challenging, since converting that obtained text mainly produced by people and arranging it into some structure information in order to analyze it with a certain machine is a very complex task.

In recent years, text analytics methods and natural language processing have become a lot more useful and accessible for every data scientist, developer, and data analyst. Today, you have that huge amount of available resources, APIs, code libraries and various services that are of great importance on your journey towards creating your first natural language processing project. In order to make it easier, text analytics project in Python will be divided into several steps, so you will better understand the overall text analytics process. You will start from scratch while being introduced to very common code libraries and APIs for every task including building a corpus, analyzing text and visualizing your results.

The step of building a corpus will be executed using Tweepy in order to gather some sample data from common Twitter's API notably used in our project for text analytics purposes. The next step includes analyzing the certain sentiment or piece of text using SDK, commonly used software development kit. The last step includes visualizing results where you will get to know how to use Pandas as well as matplotlib in order to see the final results of your project.

Building a Corpus

You are able to build your own corpus easily from everywhere if you have that large collection of e-mails that you want to analyze. You may have a collection of various customer feedbacks obtained in some surveys and you want to dive into them. On the other hand, you maybe want to devote to the voice of online customers. There are many options entirely opened to you, but in the following section, we will use some Twitter posts as a focus of our corpus. Twitter is very useful when it comes to the amazing source of different textual content since it is easily accessible as well as the public. It also offers an insight into an amazingly huge volume of text, which contains worldwide public opinion. In order to start you have to access the Twitter Search API, and doing so in Python is not that challenging. There are many libraries available to anyone, but we will use Tweepy. We will use this certain library in order to ask the Twitter for the most recent Tweets, which contain your specific search term. The following step is to write the results in a text file and each Tweet will be represented on its very own line. This will make it very easy to analyze different Tweets entirely separately in the following step. In order to begin, you will install Tweepy via pip. As soon as the installation is over, you will open Python.

```
pip install tweepy

import tweepy
```

It should be noted that you would need a permission from Twitter in order to gather the most recent Tweets. In order to do so, you will sign into Twitter as a developer in order to access tokens and get your customer keys. This process will not take long, only about three to four minutes. The following step is to build search query while adding the certain search term to that query field. You will also add some additional parameters like the amount of results you need, language, and the certain time period for search. You can get very specific about what information you need and what you need to search.

```
import the libraries
import tweepy, codecs
consumer _ key = 'your consumer key here'
consumer _ secret = 'your consumer secret key here'
access _ token = 'your access token here'
access _ token _ secret = 'your access token secret here'
auth = tweepy . OAuthHandler (consumer _ key, consumer _
secret)
auth . set _ access _ token (access _ token, access _ token _ secret)
api = tweepy . API (auth)
results = api . search(q = "your search term here", lang = "en",
result_type = "recent", count = 1000)
file = codecs . open ("your text file name here.txt", "w", "utf-8")
file . write (result . text) file . write ("\n")
file . close ()
```

The first step is to import libraries and fill in certain Twitter credentials. Then, you will let Tweepy set up a certain instance of the API and then you will fill your search query as well as store results in a certain variable. The following step is to use the codecs library in order to write the certain text of the obtained Tweets and covert it to text format. You will see that the script containing certain writing results will not be that simple API notably returning to you. In this scenario, APIs commonly return certain language data form various social media as well as from online journalism sites lots of various metadata in addition to your results. In order to do this, APIs commonly format their outputs in an easy machine to read, JSON.

In your script, every result will have its own JSON object containing text notably just a single field containing the Tweet text. When it comes to the other fields in that certain JSON file, they will contain metadata such as the timestamp or location of the certain Tweet that you easily can extract in order to get more detailed text analysis. In order to access the rest of metadata, you will need to write a JSON file. On the other hand, if you want to the full list containing JSON files, you will have to print everything and return API back to you.

Analyzing Sentiment

The following step once you collected certain text within Tweets is to analyze every single obtained Tweet. In order to start, you may use some more advanced natural language processing tools. In this certain case, we will use sentiment analysis that is very useful whether you need to obtain some positive and negative opinions or some neutral sentiment in the text you have obtained. In order to perform sentiment analysis, you will use your own AYLEIN Text API just as you used Twitter Search API. In order to begin, you have to sign up for that free plan in order to grab your API key. In this case, free means you will get that key for free permanently. This certain free plan will give you thousand calls to the API per every month entirely free of charge. In order to start, you will install using pip again, just like in the previous step.

pip install aylien - apiclient

from aylienapiclient import textapi

As soon as you get your API key, you will return your JSON results back to your metadata just as you did with that Twitter API. In order to analyze your corpus from the previous step, you have to analyze every single Tweet separately. This will give you the text containing all the Tweets predicted by your AYLIEN API. You are able to look through your file in order to verify the results as well as to visualize them to see more metrics regarding people’s feelings about your search query.

from aylienapiclient import textapi

import csv, io

```
client = textapi . Client (“your _ app_ID”, “your _ app _ key”)
io . open (‘Trump_Tweets.csv’, ‘w’, encoding=‘utf8’, newline=”)
csv _ writer = csv . writer ( csvfile )
csv _ writer ( “Tweet”, ” Sentiment” )
io . open ( “Trump.txt”, ‘r’, encoding=‘utf8’ )
f . readlines ()
tweet = tweet . strip ()
len (tweet) = = 0: print ( ‘skipped’ )
print ( tweet )
sentiment = client . Sentiment ({ ‘text’: tweet })
csv _ writer . writerow ([ sentiment[ ‘text’ ], sentiment [‘ polarity
’]])
```

The first step is to initialize that new client of your AYLIEN Text API. Then, you will remove those extra spaces that are empty so you will not waste your API credits. You will make a call to your AYLIEN Text API and write your sentiment results into csv file.

Visualizing Your Results

The next step involves visualizing your results. So far you have used an API in order to obtain that text from Twitter and you used your Text Analysis API in order to analyze what people’s opinion on a different subject is whether positive or negative in their Tweets. At this certain point, you have several options regarding what you may do with your final results. You may feed that structured information about various sentiments into some solution you are building that could be anything from an automated report to some simple social listening application. It also may be the public reaction to a certain public campaign.

You may use the obtained information in order to build some very informative visualization, and in this section, we will do just that. You will use matplotlib in order to visualize your data. You will also use Pandas in order to read your csv file. We will use this

Python library since they are very easy to get up and running. You will create visualization directly from the command line and save it as a png file.

```
pip install matplotlib
pip install pandas
import matplotlib . pyplot as plt
import pandas as pd
from collections import Counter
import
df = pd . read _ csv( csvfile )
sent = df [ “Sentiment” ]
counter = Counter ( sent )
positive = counter [ ‘positive’ ]
negative = counter [ ‘negative’ ]
neutral = counter[ ‘neutral’ ]
labels = ‘Positive’, ‘Negative’, ‘Neutral’
sizes = [ positive, negative, neutral ]
colors = [ ‘green’, ‘red’, ‘grey’ ]
yourtext = “Your Search Query from Step 2”
plt . pie (sizes, labels = labels, colors = colors, shadow = True,
startangle = 90)
plt . title ( “Sentiment of Tweets about “+ yourtext )
plt . show ()
```

You will notice that script below will open your csv file and then it will use Pandas in order to read that column titled as sentiment. It will also use Counter feature in order to count how many times every sentiment appears. Further, matplotlib will plot certain Counter’s results to some color-coded pie chart. For this step, you will have to use your text query variable mainly for presentation reasons. The first step when it comes to the visualizing your results is to import two Python libraries. Then, you will open your csv file containing the sentiment results.

Further, you will use Counter in order to count how many times a certain sentiment appears and then you will save every variable. The following step is to declare the variable in order to use them in a pie chart. In order to do so, you will use the Counter variables for different sizes. In order to plot the chart, you will use matplotlib. If you want to save your chart instead of just showing it, you will replace a part show command on that last line with another command save fig that will contain the name of your pie chart.

Chapter 2 Named Entity Extraction

Named entity extraction is the process of recognizing different named entities by identifying variously named text figures commonly including places, products, people, brands, and organizations. However, named entity extraction may also be configured to anything your organization may require like specific abbreviations, certain strains of the disease, trading stocks and almost everything else notably tagged and identified as an entity. Text analytics methods commonly identify different pattern-based entities like phone numbers, e-mail addresses, and street addresses.

In the following section of the book, you will see how to build your own named entity recognizer using Python. It is the first step on your journey towards information extraction from obtained unstructured data. It means extracting what is a real world person, organization or some other entity. You will need to map it against some knowledge base in order to understand what the certain sentence is all about. You will also need to extract certain relationships existing between different named entities. In this certain situation, we will use Python's NLTK library since it contains a standard named entity annotator, so you can get started quickly.

```
from nltk import word _ tokenize, pos _ tag, ne _ chunk
sentence = "John and Lisa are working at Google."
print ne _ chunk (pos _ tag (word _ tokenize (sentence)))
""" (S (PERSON John/NNP) and/CC (PERSON Lisa/NNP) are
/VBP working/ VBG at /IN (ORGANIZATION Google/NNP) ./.)
"""
```

This ne chink function will need part-of-speech annotations in order to add certain NE labels to specific sentences. It will further act as a chunker. It means that is produced two-level tress including nodes on the first level and nodes on the second level. Nodes within the first level are outside of that chunk while nodes on the second level are within that chunk so the bales will be denoted by that certain sub-tree.

It is very likely that for every question, there is written the answer somewhere, so you want to extract relevant and valuable information from text. It should be noted that the amount of valuable natural language text is completely astonishing and all if it is available in certain electronic form. This amount is also increasing every day. However, this great complexity of natural language can easily make it very challenging to access that information in some certain text. The state of the art when it comes to the natural language processing is still a long way from being something easily achievable and far away from being able to build those general-purposes representations of some meaning obtained from unstructured text. However, if we focus our efforts on that limited set of questions also knows as certain entity relationships we are able to significantly progress.

Groningen Meaning Bank Corpus

Named entity extraction is all about recognizing different means of unstructured data while identifying text figures including places, people, organizations, street addresses and almost everything else notably in textual form. For processing valuable information in Python, it is great to use IOB tagging that comes very useful for extracting those needed information notably used for text analytics purposes. It should be noted that IOB tagging is not that standard way of chunks since it will contain a certain initial word contained in the chunk, all words within the chunk as well as all the words outside of the chunk. Sometimes it is used a variation of IOB tagging in order to simply merge the words at the beginning of the chink and the words within of the chunk. You will have to convert your nltk tree into IOB format.

```
from nltk . chunk import conlltags2tree, tree2conlltags
sentence =  " John and Lisa are working at Google. "
```

ne _ tree = ne _ chunk (pos _ tag (word _ tokenize (sentence)))

iob _ tagged = tree2conlltags (ne _ tree)

print iob _ tagged ”’” [(‘John’, ‘NNP’, u’B-PERSON’), (‘and’, ‘CC’, u’O’), (‘John’, ‘NNP’, u’B-PERSON’), (‘are’, ‘VBP’, u’O’), (‘working’, ‘VBG’, u’O’), (‘at’, ‘IN’, u’O’), (‘Google’, ‘NNP’, u’B-ORGANIZATION’), (‘.’, ‘.’, u’O’)] ”’”

ne _ tree = conlltags2tree (iob_tagged)

print ne _ tree ”’” (S (PERSON John/NNP) and/CC (PERSON Lisa/ NNP) are/ VBP working/ VBG at/ IN (ORGANIZATION Google/NNP) ./.) ”’”

It should be noted that NLTK does not have that proper English corpus when it comes to the named entity extraction. It has only corpuses for Dutch and Spanish. You can easily try that method that will be presented here since NLTK provides an amazing corpus for readers.

In this section, we will go with GMB or Groningen Meaning Bank. It is fairly large corpus containing a huge number of annotations, but it is not perfect. It is not gold when it comes to the standard corpuses since it is not entirely human annotated, so it is not considered to be a hundred percent correct. This corpus is mainly created using some already existing annotators notably converted by humans where required. In order to start, you will download the 2.2.0 version of this corpus.

As soon as you are done, you will see that this corpus is composed of an amazingly large amount of files in tags file. It will look very messy to you, but, in fact, the data is pretty structured. You will notice that a downloaded file will contain more different sentences that are separated by two newline characters. It should be noted that for every sentence, each word would be separated by one newline character. In addition, for each word, every annotation will be separated by a certain tab character.

import os

```
import collections
ner _ tags = collections . Counter ()
corpus _ root = " gmb-2.2.0 "
filename . endswith ( ".tags" )
file _ content = file _ handle . read () . decode( 'utf-8' ) . strip ()
annotated _ sentences = file _ content . split ( '\n\n' )
annotated _ tokens = [ seq for seq in annotated _ sentence . split (
'\n' ) if seq]
standard _form _ tokens = []
annotations = annotated _ token . split ('\t')
ner _ tags [ ner ] += 1
print ner _ tags
""" Counter ({ u'O': 1146068, u'geo-nam': 58388, u'org-nam':
48034, u'per-nam': 23790, u'gpe-nam': 20680, u'tim-dat': 12786,
u'tim-dow': 11404, u'per-tit': 9800, u'per-fam': 8152, u'tim-yoc':
5290, u'tim-moy': 4262, u'per-giv': 2413, u'tim-clo': 891, u'art-
nam': 866, u'eve-nam': 602, u'nat-nam': 300, u'tim-nam': 146,
u'eve-ord': 107, u'per-ini': 60, u'org-leg': 60, u'per-ord': 38,
u'tim-dom': 10, u'per-mid': 1, u'art-add': 1 }) """
```

It should be noted that you have to make sure that your set is that proper path to that unzipped corpus. Further, you will split your sentences as well as annotations. The following step is to interpret your tags a little bit. You will notice the composition of your tags as complete tags and sub-tags. Therefore you have org tag suitable for an organization, time for time indicator, per is equal to a person, geo is equal to a geographical entity, art equals artifact, eve corresponds to an event and not equals to some natural phenomenon. It should be noted that these sub-categories are greatly polluted so they are, in fact, unnecessary. In the following step, we will remove those sub-categories in order to focus only on main categories. In order to do so, you have to modify the code a little bit.

```
ner _ tags = collections . Counter ()
```

```
filename . endswith ( ".tags" )
file _ content = file _ handle . read () . decode ( 'utf-8' ) . strip ()
annotated _ sentences = file _ content . split ( '\n\n' )
annotated _ tokens = [ seq for seq in annotated _ sentence . split (
'\n' )
standard _form _ tokens = [ ]
annotations = annotated _ token . split ( '\t' )
ner = annotations [ 0 ], annotations [ 1 ], annotations [ 3 ]
ner != ' O '
ner = ner . split ( '-' ) [ 0 ]
ner _ tags [ ner ] += 1
print ner _ tags
Counter ({ u'O': 1146068, u'geo': 58388, u'org': 48094, u'per':
44254, u'tim': 34789, u'gpe': 20680, u'art': 867, u'eve': 709,
u'nat': 300 })
print "Words=", sum ( ner _ tags . values ( ))
Words= 1354149
```

You will notice that this looks much better and less polluted. You also have an option to drop that last few tags in the case when they are not well represented in the corpus.

Training Your System

When it comes to the training your system, we will use very similar approach to one commonly used for training a part-of-speech tagger. In order to begin, you have to import string and init that stemmer. The following step is to pad the sequence with certain placeholders and shift that index with two in order to accommodate the padding. This feature extraction will work just like features working as the part of part-of-speech tagger training except for added history mechanism. It should be noted that IOB tag is an amazingly good indicator or what your current IOB tag will be, so you will include that previous IOB tag as a mean feature.

```
import string
from nltk . stem . snowball import SnowballStemmer
def features ( tokens, index, history ):
””” `tokens` = a POS-tagged sentence [( w1, t1 ), ...]
`index` = the index of the token we want to extract features for
`history` = the previous predicted IOB tags ”””
stemmer = SnowballStemmer ( ‘english’ )
tokens = [( ‘[ START2 ]’, ‘[START2]’), (‘[ START1 ]’, ‘[ START1 ]’)] + list ( tokens ) + [(‘[ END1 ]’, ‘[ END1 ]’), (‘[ END2 ]’, ‘[END2 ]’)]
history = [‘[ START2 ]’, ‘[ START1 ]’] + list ( history )
index += 2 word
pos = tokens [ index ]
prevword, prevpos = tokens [ index – 1 ]
prevprevword, prevprevpos = tokens [ index – 2 ]
nextword, nextpos = tokens [ index + 1 ]
nextnextword, nextnextpos = tokens [ index + 2 ]
previob = history [ index – 1 ]
contains _ dash = ‘-‘ in word
contains _ dot = ‘.’ in word
allascii = all ([ True for c in word if c in string . ascii _ lowercase ])
allcaps = word = = word . capitalize ()
capitalized = word [ 0 ] in string . ascii _ uppercase
prevallcaps = prevword = = prevword . capitalize ()
prevcapitalized = prevword [ 0 ] in string . ascii _ uppercase
nextallcaps = prevword = = prevword . capitalize ()
```

```
nextcapitalized = prevword [ 0 ] in string . ascii _ uppercase
return
{ ‘word’: word, ‘lemma’: stemmer . stem (word),
‘pos’: pos,
‘all - ascii’: allascii,
‘next - word’: nextword,
‘next - lemma’: stemmer . stem ( nextword ),
‘next - pos’: nextpos,
‘next – next - word’: nextnextword,
‘nextnextpos’: nextnextpos,
‘prev - word’: prevword,
‘prev - lemma’: stemmer . stem ( prevword ),
‘prev - pos’: prevpos,
‘prev - prev - word’: prevprevword,
‘prev – prev - pos’: prevprevpos,
‘prev-iob’: previob,
‘contains-dash’: contains _ dash,
‘contains-dot’: contains _ dot,
‘all - caps’: allcaps,
‘capitalized’: capitalized,
‘prev – all - caps’: prevallcaps,
‘prev-capitalized’: prevcapitalized,
‘next – all - caps’: nextallcaps,
‘next - capitalized’: nextcapitalized, }
```

The following step is to create the utility functions in order to make the training process easier as well as to move that corpus reading stuff into a certain read gmb function. You will manage to read your sentences from that corpus in some proper format. The following step is to actually start training your system and NLTK provides an amazing amount of very helpful classes in order to help you train your system. You will use a Naive Bayes classifier in order to predict certain sequences.

```
def to _ conll _ iob ( annotated _ sentence )
proper _ iob _ tokens = [ ]
ner = annotated _ token
ner != ‘ O ‘
idx = = 0:
ner = “B-” + ner elif annotated _ sentence [ idx – 1 ] [ 2 ] = = ner
ner = “I-” + ner
ner = “B-” + ner
proper _ iob _ tokens . append (( tag, word, ner ))
return proper _ iob _ tokens
def read _ gmb ( corpus _ root )
filename . endswith ( “.tags” )
file _ content = file _ handle . read ( ) . decode ( ‘utf-8’ ) . strip ( )
annotated _ sentences = file _ content . split ( ‘\n\n’ )
annotated _ tokens = [ seq for seq in annotated _ sentence . split (
‘\n’ ) if seq ]
standard _form _ tokens = [ ]
annotations = annotated _ token . split ( ‘\t’ )
ner = annotations [ 0 ], annotations [ 1 ], annotations [ 3 ]
ner != ‘O’:
ner = ner . split ( ‘-‘ ) [ 0 ]
tag = “““”
standard _form _ tokens . append (( word, tag, ner ))
conll _ tokens = to _ conll _ iob ( standard _form _ tokens )
yield [(( w, t ), iob ) for w, t, iob in conll _ tokens ]
reader = read _ gmb ( corpus _ root )
```

When it comes to the training your system, you will use a certain annotated sentence containing a list of triplets including person and location. The following step is to make your NLTK entirely compatible to that NLTK classifier since the classifiers will expect a tuple to be the initial item input. As soon as you done, you will, check the output and you will see that you are able to read your sentences from your corpus easily in a proper format. In the following steps, you will transform your results to some preferred list containing triplet format. The next step is to transform that list containing triplets into NLTK.

```
import pickle
from collections import Iterable
from nltk . tag import ClassifierBasedTagger
from nltk . chunk import ChunkParserI
class NamedEntityChunker ( ChunkParserI )
def __ init __ ( self, train _ sents, **kwargs )
assert isinstance ( train _ sents, Iterable )
self . feature _ detector = features
self.tagger = ClassifierBasedTagger ( train=train _ sents, feature _
detector=features, **kwargs )
chunks = self . tagger . tag ( tagged _ sent )
iob _ triplets = [ ( w, t, c ) for ( ( w, t ), c ) in chunks ]
return conlltags2tree ( iob _ triplets )
```

The following step is to build the dataset, so your system will be able to recognize different named entities like in this case time and geographical entity. You also can test your system since you followed this good pattern in NLTK.

Chapter 3 Getting Started with NLTK

It is very easy to get your hands on millions of words contained in textual format. So, you wonder what can you do with that. You have numerous options available by combining simple programming techniques using Python that can easily handle large portions of unstructured text. You can automatically extract some keywords as well as various phrases notably summing the content and style of the text you have obtained. Python provides many different tools and kits in order to perform such action like Natural Language Processing kit or simply NLTK. In this section of the book, you will get to know NLTK and the concept behind it. You will use some raw data for your program that will manipulate that data in a variety of different as well as exciting ways. Before you embark on this adventure, you have to get started with certain Python interpreter.

There are numerous friendly things about Python especially since it allows you to type some words directly into Python interactive interpreter. It is a certain programme that will run all of your Python programmes. You are able to access it using a very simple graphical interface IDLE or Interactive DeveLopment Environment. Under Unix, you are able to run it from the shell just by typing idle. As soon as you are don, the interpreter will print everything about your current Python version.

Python 3.4.2 (default, Oct 15 2014, 22:01:37)

[GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.40)] on darwin

Type "help", "copyright", "credits" or "license" for more information

If you are unable to run the program, it is because you do not have Python installed at all. In order to get your Python version, just visit Python website, which provides all details about latest NLTK versions notably working excellently on the latest Python versions.

In the case when you already installed the program, you will see that it automatically indicates the Python interpreter waiting for some new input. In order to test it, you can do some calculations so Python will be used as a calculator. As soon as you are done, you will see that the prompt reappear displaying your answer. In order to do some more calculations, you simply enter more expressions of your own using an asterisk and other commands. In the next section, you will see the preceding exampled notably demonstrating how you can work very interactively with this Python interpreter. You are able to experiment with various expressions in the language and you will see what the program will do with them.

In the case when you try some nonsensical expression, the Python interpreter will display a syntax error. In Python, it really does not make any sense if you end your expression with certain plus sign. The program will automatically indicate a certain line when the issue occurred. As soon as you are ready with NLTK, you are able to work with your language data. In order to start, you will install latest NLTK version and by following provided instructions, you will be ready in almost no time. Once you have your NLTK version, you will use the Python interpreter just like in the previous example. You will install all required data for your task and then type commands at the interpreter.

```
import nltk
nltk . download ()
from nltk.book import
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it
Type: 'texts ()' or 'sents ()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: Inaugural Address Corpus
text4: Chat Corpus
text5: Monty Python and the Holy Grail
```

text6: The Man Who Was Thursday by G . K . Chesterton 1908

Once all the required data is downloaded, you are able to load some data into the interpreter. The initial step is to type some special command at the prompt that tells the interpreter to load your text so you can further explore it using NLTK. You will use command from nltk import. This certain command tells the program to load certain textual data. The certain book module will contain all the data needed in order to process it further using the program. As soon as you start, you will see that welcome message and your text will be loaded. Make sure that you check your spelling and punctuation. Anytime you want to find out more about some textual data, you will type the names directly at the Python interpreter.

Searching Syntax

When it comes to the searching text or syntax in order to examine the content of your text, there are many ways to so, so you have options besides just reading your text. In order to display the content, you will use a concordance view notably showing you every occurrence of a certain word together with the context. For instance, you can search for a certain word in some book. In the following section, we will search for word monstrous in the Moby Dick book. In order to do so, you will enter text1 command followed by the term concordance. You will place word monstrous in the command like as well after you entered the word concordance. The very first time when you use this term concordance, the overall process may take up to few minutes since the program has to take some time in order to build a certain index containing your subsequent searches.

text1 . concordance("monstrous")

Displaying matches:

One was of a most monstrous size

Touching that monstrous bulk of the whale or ork

Heathenish array of monstrous clubs and spears

Some were thick d as you gazed, and wondered what monstrous cannibal and savage could ever have that has survived the flood

Most monstrous and most mountainous

In connection with the monstrous pictures of whales

I am strongly here to enter upon those still more monstrous stories of them

Out of this monstrous cabinet

Whales of a monstrous size…

You have an option to search for any word you like. When it comes to the saving your typing, you can use different commands in order to access your previous command in order to further modify the word that you have searched previously. You can try searches on some other books, articles and pretty much everything else that are included in NLTK and the process will be the same like in this previous example using command concordance. Once you have spent more time examining your text, you will get that sense of richness as well as that great diversity of language. This command concordance will allow you to see words in some context. You have an option to find out more by appending another term similar to that word you are searching.

text1 . similar ("monstrous")

text2 . similar ("monstrous")

You will notice that you got the different results when it comes to the different texts. You will notice that other words will appear in a very similar range of contexts and you will find those contexts by typing similar in the command line. Further, you will insert some relevant words that you are looking for in parentheses. You also have an option to use another term, common context notably allowing you to examine only a single context, which is shared by two or more words like very and monstrous. In order to do so, you will enclose your words in square brackets and further separate them using a comma.

text2.common _ contexts ([“monstrous”, “very”])

a _ pretty is _ pretty am _ glad be _ glad a _ lucky

You may pick another pair of words and further compare their usage in different texts using that common context and similar functions. These functions automatically will detect your particular word occurring in some obtained textual data. Further, it will display those words you have searched for and their occurrence within a certain context. This function also can easily determine the location of some word in the text. The function also may show you how many times that same word appears in some context. These location related information will be displayed using command dispersion plot. It will represent every stripe regarding the instance of a certain word while every row will represent the complete text. You are able to produce different plots using a dispersion plot displaying information regarding the location and position of the text. You can type various words and different texts, just make sure that you use brackets, quotes, parentheses, and commas correctly.

text4 . dispersion _plot ([“citizens”, “democracy”, “freedom”, “liberty”, “world”])

text3 . generate ( )

It should be noted that you would Matplotlib and NumPy packages are installed to produce some graphical plots. You also have an option to plot the frequency of some word through time. The last step when it comes to the searching syntax is to generate your text in the different styles. In order to do so, you will just enter the name of the text and add term generate. It should be noted that you have to include parentheses, but you will not separate them with commas.

Sentence Tokenization

NLTK is a leading platform when it comes to the natural language processing in Python. It is designed in order to work with human language while providing easy-to-use interface containing more than fifty corpora and different lexical resources like WordNet an addition to a suite of various text processing libraries used for classification, tagging, parsing, tokenization, semantic reasoning and everything else needed to embark on this text analytics journey.

NLTK has proven to be an amazing tool for working in and learning using Python containing an amazing library used to play with natural language. Thanks so very comprehensive API documentation and provided computational linguistics, educators, students, researchers, and analytics gladly use NLTK when it comes to the text analytics tasks. Due to these numerous advantages, we will use NLTK in order to tokenize text and words. We will use NLTK tokenizer that will divide a string into some sub-strings by dividing that specified string. In lexical meaning, it is a lexical analysis of textual content meaning you will be splitting your text into a sentence. Even though it seems like a very simple task, it may be challenging.

NLTK tokenizer is designed in order to be very flexible as well as very easy to adapt to some new domains and challenging tasks. The concept lying behind simple NLTK tokenizer is that tuple regex string notably defining a list of some regular expression strings. These strings are into certain order containing a compiled list of various regular expression objects we call words.

The overall process of tokenization will be performed using certain word re-findall. In this situation, the letter S corresponds to the certain user-supplied string contained inside that certain tokenizer. When you are installing tokenizer objects, you will have a single command known as preserve case. By default, this option is set to be True. On the other hand, if you set it to be false, your tokenizer immediately down-case everything but emoticons.

span _ tokenize (s) [source]

tokenize (s)

You have an option to use this tokenizer feature for Tweets called as Tweet Tokenizer. You will use the base set as different objects by preserve case command that will set as default True. The first step is to import Tweet Tokenizer.

from nltk . tokenize import TweetTokenizer

tknzr = TweetTokenizer ( )

s0 = " This is a cooool # dummysmiley: :-) :-P <3 and some arrows < > -> <—"

tknzr . tokenize (s0) [' This ', ' is ', ' a ', ' cooool ', '# dummysmiley', ':', ':-)', ':-P', '<3', 'and', 'some', 'arrows']

tknzr = TweetTokenizer (strip _ handles = True, reduce _ len=True)

s1 = '@remy: This is way too much for you!!!!!!'

tknzr . tokenize (s1) [':', 'This', 'is', 'way', 'too', 'much', 'for', 'you', '!', '!', '!']

When it comes to the parameters that you will use, the first one is text and return command. Returns parameter represents a tokenized list of different strings. This list will be returned to that original list if you use command preserve case and set it to False. You will also use convenience function when it comes to the wrapping your tokenizer. By lexical analysis or sentence tokenization, you will convert a sequence of different characters contained in textual format into certain sequences of tokens. These tokens represent certain strings that come with already assigned and identified meaning. The program used for sentence tokenization, in this case, NLTK will analyze the syntax of textual data.

We will use sentence tokenizer in order to find a certain list of various sentences. You can also find different Word tokenizer that you can further use in order to find strings contained in certain words. Tokenizing text into specific sentences is also known as sentence segmentation, sentence boundary disambiguation or sentence boundary detection known as sentence-breaking.

Sentence boundary disambiguation or sentence breaking is among the main problems when it comes to the natural language processing. It is the process of deciding where certain sentences begin and where they end. It is common for natural language processing tools to require their input to be previously divided into certain sentences for many reasons.

On the other hand, sentence boundary disambiguation is very challenging since punctuation marks are greatly ambiguous. For instance, a period may denote decimal point or abbreviation, but not the end of a sentence. There are many tools available when it comes to the sentence tokenization like NLTK, which will use. In order to begin, you will install nltk data and launch the program. You will import sentence tokenization tool from NLTK.

```
text = this's a sent tokenize test.
from nltk . tokenize import sent _ tokenize
sent _ tokenize _ list = sent _ tokenize ( text )
len ( sent _ tokenize _ list )
sent _ tokenize _ list [ " this's a sent tokenize test. " ]
```

You will use a sent tokenize test after you have imported from NLTK tokenization tool. You will use command sent tokenize from the NLTK. It should be noted that this instance has already been trained and it will work for almost every European language. It already knows what characters and punctuation marks end sentences and what the beginning of a new sentence looks like. You will use sent command sent tokenize of Punkt Sentence Tokenizer from that nltk tokenize command. It should be noted that this Tokenize Punkt has been already trained model used for the majority of European languages. There are seventeen languages supported by NLTK when it comes to the sentence tokenization, and you are able of using all of them.

```
import nltk . data
tokenizer = nltk . data . load ( ‘ tokenizers / punkt /English . pickle’ )
tokenizer . tokenize ( text ) [ “this’s a sent tokenize test.” , ‘ this is sent two.’ , ‘is this sent three?’ , ‘]
```

A spanish sentence tokenize example:

```
spanish _ tokenizer = nltk . data . load ( ‘ tokenizers/ punkt/ spanish . pickle’ )
spanish _ tokenizer . tokenize ( ‘Hola amigo. Estoy bien.’ ) [ ‘Hola amigo.’, ‘Estoy bien.’ ]
```

When it comes to the tokenizing some text into word using Python’s NLTK, the process is very simple. You will only have to use command word tokenize from that previous nltk tokenize module. It should be noted that this command word tokenizer, in fact, is a simple wrapper function notably calling tokenize.

```
from nltk . tokenize import word _ tokenize
word _ tokenize ( ‘ Hello World. ’ ) [ ‘ Hello ’, ‘ World ’, ‘.’ ]
word _ tokenize ( “ this’s a test ” ) [ ‘ this ’, “ ‘s ”, ‘ a ’, ‘test’ ]
```

Standard word tokenizer:

```
_word _ tokenize = TreebankWordTokenizer ( ) . tokenize
def word _ tokenize ( text )
return _ word _ tokenize( text )
```

In this example, you have seen that standard work tokenizer just returns that tokenized copy of your text since it is designed in order to work on a single sentence at a time. However, there is another equivalent method you can use in order to provide the same result. You also have an option to use different word tokenizers in addition to what I have used here Word Punkt Tokenizer. You may use

Punkt Tokenizer that splits on punctuation but at the same time keeps it within the word.

```
from nltk . tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer ( )
tokenizer . tokenize ( “ this’s a test ” ) [ ‘ this ’, “‘ s ”, ‘ a ’, ‘ test ’
]
from nltk . tokenize import PunktWordTokenizer
punkt _ word _ tokenizer = PunktWordTokenizer ( )
punkt _ word _ tokenizer . tokenize ( “ this’s a test ” ) [ ‘ this ’, “‘ s
”, ‘ a ’, ‘ test ’ ]
```

You have an option to choose any word tokenizer your like from NLTK depending on your using purpose. In the following chapter, you will how to use NLTK in order to automatically summarize any text you have obtained.

Chapter 4 Automatic Text Summarization

Automatic text summarization is common text analytics process of reducing some text document using a computer program to create a summary, which will retain only the most important points of that original document. Since there is a common problem when it comes to the overloading of information since it has grown rapidly, the quantity of data has rapidly increased. Therefore, we simply need a program that will perform an automatic text summarization. Fortunately, technologies available can easily make that coherent summary taking into account different variables like writing style, syntax, and length of some text document. In order to start, you will import NLTK summarizer.

```
from summa import summarizer
print summarizer . summarize ( text )
```

The next step is to extract keywords. Like I already stated , text summarization process is all about generating summaries from some given long text. We will use an approach of extracting sentences, which consist of some more frequent words, so the overall process of text summarization may be easily performed just by extracting a few certain sentences.

```
from nltk . tokenize import sent _ tokenize, word _ tokenize
from nltk . corpus import stopwords
from collections import defaultdict
from string import punctuation
from heapq import nlargest
class Summarize _ Frequency: def __ init __ ( self, cut _ min=0.2,
cut _ max=0.8):
```

You will use function class summarize frequency and then you will initialize certain text summarizer. It should be noted that words, which have certain frequency term lower than function cut min or higher than function cut max will be entirely ignored. When it comes to the keyword extraction, you will use function print keywords.

```
from summa import keywords
print keywords . keywords ( text )
```

Your installed software will depend on Scipy and NumPy, two packages commonly used for scientific computing. It should be noted that you have to install them before you begin a process of text summarization. If you are going to use a certain export function, then you will install NeworkX, so you will install Pattern in order to enhance keyword extraction process. In the following section, you will learn how to use command-line, how to export and how to define the length of your summary corresponding to a certain proportion of your text which is also available in keywords.

```
cd path/ to/ folder/ summa/ python textrank.py -t FILE
from summa . export import gexf _ export
gexf _ export ( text, path = “graph . gexf ” )
from summa . summarizer import summarize
summarize ( text, ratio = 0.2 )
summarize ( text, words=50 )
summarize ( text, language = ‘spanish’ )
summarize ( text, split = True )
```

You also have an option to summarize your text by defining a length of your summary using an approximate number of contained words. You also have to define the input language available in keywords. The final step is to get your results using summarize function.

Chapter 5 Text Classification Using Scikit-Learn and NLTK

In this chapter, you will get to know how to use popular Python library scikit-learn in order to classify certain text. It should be noted that document text classification is one of the most important as well as the most common task when it comes to the text analytics tasks. Assigning certain categories to textual data that can be a library book, web page, gallery or media articles has various applications like email routing, spam filtering, and sentiment analysis.

Before you embark on text classification using Python's scikit-learn we will divide this task into certain steps. The first step will be to set up a certain environment and load the relevant data set in jupyter. The following step requires extracting useful features from your text files and running ML models. The next step includes Grid Search in order to perform parameter tuning.

For your first step of setting up a certain environment, you will have to provide Python version 2.7.3. in addition to the jupyter notebook. You will have to install anaconda. It will not be a problem to execute this test since you already know the fundamentals behind text analytics. For text classification, you will use scikit-learn in addition to a little bit of NLTK. The second step includes loading the data set, and for this step, you can use any set you previously obtained.

In this section, I will show you how to use 20 Newsgroup data collection containing around 20,000 newsgroup documents notably partitioned evenly across twenty different newsgroups. This certain collection was introduced by Ken Lang. It has become very popular data set when it comes to the various text analytics experiments using various machine learning techniques like text clustering and text classification.

It should be noted that this data collection is already built in scikit-learn, so you can begin immediately. You will open a command prompt and just type jupyter notebook. Further, you will open the notebook and your session will immediately start. You will select Python 2 and give a certain name to the obtained jupyter notebook. The next step is to load the data collection which will take up to few minutes.

```
from sklearn . datasets import fetch _ 20newsgroups
twenty _ train = fetch _ 20newsgroups ( subset= ‘ train ‘, shuffle = True )
```

It should be noted that you will load the test data entirely separate from the training data. The next step requires checking the categoric or target named and data files and you will print all available categories while using command notably printing an initial line of your first data file.

```
twenty _ train . target _ names
print ( ” \ n ” . join ( twenty _ train . data [ 0 ] . split ( ” \ n ” ) [ :3 ] ))
```

The following step requires extracting valuable features from your text file. A text file is only series containing words in order and to run machine learning model you will have to convert a certain text file into specific numerical feature vectors. In this case, you will use a bag of words algorithm. As soon as you obtain the bag of words, you will segment every text file into certain words splitting them using space. You also have to count how many times every word occurs in every text document and then you will assign every word an integer. It should be noted that every word contained in your dictionary will correspond to certain descriptive feature. Fortunately, scikit-learn has a very high-level component that will easily create all required feature vectors using Count Vectorizer.

```
from sklearn . feature _ extraction . text import CountVectorizer
```

```
count _ vect = CountVectorizer ( )
X _ train _ counts = count _ vect . fit _ transform ( twenty _ train .
data )
X _ train _ counts . shape
```

In this previous example by using count vector and fit transform function, you will be learning certain vocabulary dictionary notably returning your document's matrices. Further, you have to use function term frequencies that just counts the number of every word in every document since there is an issue. It will give greater weigh the to some longer documents. In order to avoid this issue, you will use term frequencies that count a total number of words in every document. The final step requires reducing that weight of some very common words like an, is or the. This generally is called as term frequency times inverse total document frequency.

```
from sklearn . feature _ extraction . text
import TfidfTransformer
tfidf _ transformer = TfidfTransformer ( )
X _ train _ tfidf = tfidf _ transformer . fit _ transform ( X _ train _
counts )
X _ train _ tfidf . shape
```

The following step required in any text classification task is running ML models. There are various models that can be used for text classification task. In this case, I will use a single Naive Bayes notably the simplest one. You can very easily build a Naive Bayes classifier using Python's scikit-learn with just two lines of codes. As soon as you are done, you will have to train your Naive Bayer classifier on your training data and then build a pipeline.

```
from sklearn . naive _ bayes import MultinomialNB
clf = MultinomialNB ( ) . fit ( X _ train _ tfidf, twenty _ train .
target )
```

```
from sklearn . pipeline import Pipeline
text _ clf = Pipeline ( [ (‘ vect ‘, CountVectorizer ( ) ), ...
( ‘tfidf’, TfidfTransformer ( ) ), ...
( ‘clf’, MultinomialNB ( ) ), ... ] )
text _ clf = text _ clf . fit ( twenty _ train . data, twenty _ train .
target )
```

It should be noted that arbitrary vectors will be used in the following steps. The next step is to calculate the overall performance of your Naive Bayer classifier on your test set. You will get an approximate accuracy and if it is above seventy percent, then consider your Naive Bayes properly working since that accuracy is not bad at all for a Naive classifier. Then, you will have to use SVMs or support vector machines if you want to check your model and try getting some better performance results. You will see that you performance accuracy is better while using support vector machines. It will probably be above eighty percent.

```
import numpy as np
twenty _ test = fetch _ 20newsgroups ( subset =’ test ‘, shuffle =
True )
predicted = text _ clf . predict ( twenty _ test . data )
np . mean ( predicted = = twenty _ test . target )
from sklearn . linear _ model import SGDClassifier
text _ clf _ svm = Pipeline ( [ ( ‘vect’ , CountVectorizer ( ) )
( ‘tfidf’, TfidfTransformer ( ) )
( ‘clf-svm’, SGDClassifier ( loss = ‘ hinge ‘, penalty = ‘ l2 ‘, ...
alpha = 1e - 3, n _ iter = 5, random _ state = 42 ) ), ... ] )
_ = text _ clf _ svm . fit ( twenty _ train . data, twenty _ train .
target )
predicted _ svm = text _ clf _ svm . predict ( twenty _ test . data )
np . mean ( predicted _ svm = = twenty _ test . target )
```

Like I already mentioned, the last step in this text classification task is grid search. It should be noted that almost every classifier will contain various parameters which easily can be tuned in order to obtain that optimal performance. Python's scikit-learn when it comes to the Grid Search is an extremely useful tool. You will create a certain list containing different parameters notably used in order to tune the overall performance of your model. Every parameters name will start with certain classifier name. The next step required is creating an instance of that grid search by passing certain classifiers and parameters. In order to do so, you will use function n jobs that tell your model to use some multiple cores from your model.

```
from sklearn . model _ selection import GridSearchCV
parameters = {'vect __ ngram _ range ' : [( 1, 1 ), ( 1, 2 )], ...
'tfidf __ use _ idf ': ( True, False ), ...
'clf __ alpha': ( 1e-2, 1e-3 ),...
gs _ clf = GridSearchCV ( text _ clf, parameters, n _jobs= - 1 )
gs _ clf = gs_clf.fit(twenty_train.data, twenty _ train . target )
gs _ clf . best _ score _ gs _ clf . best _params _
```

It should be noted that this process will take a few minutes depending on your machine configuration. The last step is to see that best mean score in addition to the params. You will notice that you have increased your overall model's performance up to ninety percent for your Naive Bayes classifier. Very similarly you can improve the overall performance of your support vector machine classifier in the same manner by tuning relevant parameters.

```
from sklearn . model _ selection import GridSearchCV
parameters _ svm = { ' vect __ ngram _ range ' : [ ( 1, 1 ), ( 1, 2 )
], ...
'tfidf __ use _ idf ': ( True, False ), ...
'clf-svm __  alpha ': ( 1e-2, 1e-3), ... }
gs _ clf _ svm = GridSearchCV (text _ clf _ svm, parameters_svm,
n _jobs =-1 )
```

gs _ clf _ svm = gs _ clf _ svm . fit (twenty_train . data, twenty _ train . target)

gs _ clf _ svm . best _ score _ gs _ clf _ svm . best _ params _

When it comes to the text classification using NLTK, one of the most important steps is removing stop words. You should perform this step when you have some stop words which are not so useful for your problem. It should be noted that removing stop words will also increase the overall accuracy of your model. In order to do so, you will have top use Count Vectorizer.

from sklearn . pipeline import Pipeline

text _ clf = Pipeline ([('vect' , CountVectorizer (stop _ words = ' english ')), ...

(' tfidf', TfidfTransformer ( )), ...

(' clf', MultinomialNB ( )), ...])

Another useful function that you can commonly use if fit prior equals to false. When set to false when it comes to the Multinomial Naive Bayes will not be of great importance. However, this function will also improve the overall accuracy, so you can try it and see if it works for your model's accuracy.

Stemming Words with NLTK

Another very important feature provided by Python's natural language processing kit NLTK is stemming words. Stemming is a very common process of reducing some derived or inflected words to their word base or word stem. For instance, a process of word stemming can reduce words like fished or fishing to their stem word that is fish. In order to perform stemming of the words, you will have to use NLTK since it comes with already built-in stemmers that will be of great help when it comes to the reducing words to their roots. In this certain case, you will use NLTK's built-in Snowball stemmer that works amazingly well for the English language.

```
import nltk
nltk . download ( )
from nltk . stem . snowball import SnowballStemmer
stemmer = SnowballStemmer ( ” english “, ignore _ stopwords = True )
class StemmedCountVectorizer ( CountVectorizer )
analyzer = super ( StemmedCountVectorizer, self ) . build _ analyzer ( )
return lambda doc:
( [ stemmer . stem ( w ) for w in analyzer ( doc ) ] )
stemmed _ count _ vect = StemmedCountVectorizer ( stop _ words = ‘ english ‘ )
text _ mnb_stemmed = Pipeline ( [ ( ‘ vect ‘, stemmed _ count _ vect ), ...
( ‘ tfidf ‘, TfidfTransformer ( ) ), ...
( ‘ mnb ‘ , MultinomialNB (fit _prior = False ) ), ... ] )
text _ mnb _ stemmed = text _ mnb _ stemmed . fit ( twenty _ train . data, twenty _ train . target )
predicted _ mnb _ stemmed = text _ mnb _ stemmed . predict ( twenty _ test . data )
np . mean ( predicted _ mnb _ stemmed = = twenty _ test . target )
```

Using NLTK Snowball stemmer, you will be able to improve your model’s accuracy up to eighty-two percent even though it will be only a marginal improvement with Naive Bayes classifier. You can use word stemmer on your other algorithms as well like support vector machines. In this chapter, you have seen how to perform text or document classification using Python’s NLTK and scikit-learn. Now, you also know what can greatly improve your model’s accuracy especially if you use already built-in word stemmer in

NTLK. You have also seen that for your data set, both of these two used algorithms equally matched while being optimized. Commonly when you have enough data set, that certain choice of different algorithms will not make any significant difference.

Topic Modeling

When it comes to the analytics industry and text analytics in generall, it is all about obtaining some useful information from some unstructured data. We all have witnessed that rapidly growing amount of data especially when it comes to these recent years. Fortunately, technology has provided and developed some very powerful techniques that can be easily used in order to mine through the data and further fetch the information that is useful and relevant and something we are looking for.

One very useful text analytics technique is topic modeling. It is commonly used process in order to automatically identify some topics notably present in some text object. Topic modeling process further derives that hidden patterns notably exhibited by some text corpus.

Topic modeling is greatly different from some rule-based text analytics techniques which use those regular expressions as well as some dictionary based keyword techniques. It should be noted that this process is an unsupervised method commonly used for finding as well as observing some bunch of words namely topics in a certain large cluster of textual data. These topics are commonly defined as some repeating pattern of that co-occurring terms in a certain text corpus.

Topic models come to be very useful for various text analytics purposes like organizing some large blocks of various textual data, document clustering, feature selection or information retrieval. For instance, some major publishing companies like New York Times are using text analytics topic models in order to boost their article recommendations systems.

On the other hand, various other professionals turn to topic modeling when it comes to the recruitment industries where they mainly aim to extract those latent features of various job descriptions a and to further map them to those right candidates. Topic models are commonly used when it comes to the organizing some large datasets like customer reviews, social media profiles or emails.

When it comes to the obtaining topics from some textual data, you can use several approaches including inverse document frequency and term frequency. However, the most popular technique used in topic modeling is Latent Dirichlet Allocation since it assumes various documents notably produced from a certain mixture of different topics. These certain topics further are generated commonly based on certain probability distribution. This approach further backtracks as well as trie to figure certain topic from obtained textual data. The approach, in fact, is a matrix factorization technique using certain vector space or any collection of documents that can be easily represented as a certain document-term matrix.

When it comes to the parameters used in Latent Dirichlet Allocation, alpha parameter represents document-topic density while beta parameter represents that topic-word density. It should be noted that higher alpha parameter means that your document is composed of a greater number of topics while a lower value of alpha means that your document is composed of the lower amount of different topics. On the other hand, if you have beta parameter higher your document contains a higher amount of words while lower beta parameter means that your document is composed of the lower amount of words.

In the following section, you will estimate the number of topics, number of certain topic terms and number of specific passes or iterations. This number of a topic will be directly extracted from your obtained corpus while a number of terms will be composed of a single topic. A number of iterations or passes will give you that maximum number of all allowed passes to your Latent Dirichlet Allocation for further convergence. In order to begin, you will import some sample documents in order to form your text corpus.

```
doc1 = ” “
doc2 = ” “
doc3 = ” “
doc4 = ” “
doc5 = ” “
doc_complete = [ doc1, doc2, doc3, doc4, doc5 ]
```

The next step is data preprocessing and data cleaning. These two steps are among the most important steps when it comes to the topic modeling where you will remove the stopwords and punctuations in order to normalize your corpus. In this step, we will use again NLTK.

```
from nltk . corpus import stopwords
from nltk . stem . wordnet import WordNetLemmatizer
import string
stop = set ( stopwords . words ( ‘ english ‘ ) )
exclude = set ( string . punctuation )
lemma = WordNetLemmatizer ( )
def clean ( doc ):
stop _free = ” “. join ( [ i for i in doc . lower ( ) . split ( ) if i not in stop ] )
punc _free = ” . join (ch for ch in stop _free if ch not in exclude )
normalized = ” ” . join ( lemma . lemmatize ( word ) for word in punc _free. split ( ) )
```

```
return normalized doc _ clean = [ clean ( doc ) . split () for doc in doc _ complete ]
```

The following step requires preparing your document-term matrix. You already know that all obtained documents combined will for a corpus. In order to run mathematical models on your text corpus, you will convert it into certain matrix representation. Python provides amazing libraries when it comes to the text analytics practices like gensim. As soon as you import library gensim, you will create that term dictionary of your corpus where every since term will be assigned an index. The next step is to convert your list of documents into a certain document-term matrix.

```
import gensim
from gensim import corpora
index . dictionary = corpora . Dictionary ( doc _ clean )
doc _ term _ matrix = [ dictionary . doc2bow ( doc ) for doc in doc _ clean ]
```

The following step of topic modeling requires running your Latent Dirichlet Allocation model. You will train your model on your document-term matrix. It should be noted that this training will require a few parameters like input. The gensim module will allow you both your model estimation from an inference of topic distribution and from training model on some new and unseen documents.

```
Lda = gensim . models . ldamodel . LdaModel
ldamodel = Lda ( doc _ term _ matrix, num _ topics = 3, id2word = dictionary, passes = 50 )
print ( ldamodel . print _ topics ( num _ topics = 3, num _ words = 3 ) ) [ '0.168* + 0.083* + 0.072*, '0.061* + 0.050* + 0.050*, '0.049* + 0.049* + 0.049* ]
```

It should be noted that every line is a certain topic within some individual weights and topic terms. Also, these results of your topic models are entirely dependant on some terms or features notably present in your corpus. Your corpus is represented entirely as document term matrix that is very spared in its nature. On the other hand, if you reduce the overall dimensionality of the matrix, you will be able to greatly improve the results of your topic modeling process. I order to do so, you can turn to frequency filter, part of speech tag filter or batch-wise LDA.

If you use frequency filter, you will arrange each term in accordance with its frequency. It should be noted that terms with some higher frequencies have a greater probability of appearing in your results as those with some lower frequencies. Also, these low-frequency terms are weak features within your corpus, so it is a great approach if you just get rid of all weak features using exploratory analysis of different teams notably helping you decide what certain frequency value may be considered as that specific threshold.

You also can use batch wise LDA to retrieve some of the most important topic terms. If you use this feature, you will divide your corpus into certain batches of some fixed sizes. If you run LDA multiple times on these specific batches, you will generate different results. It should be noted that the best batch wise LDA approach is to do the intersection of all available batches. Also, model LDA commonly is used as valuable feature selection methods especially when you have complex text analytics task where your training data contains a greater number of wise documents.

Chapter 6 Part of Speech Tagging

Part of speech tagging is among the most important text analytics tasks commonly used when it comes to the classifying various words into their specific part-of-speech. This process also labels certain words in accordance to certain tagset that is a collection containing various tags commonly used for the part-of-speech tagging. This process is also commonly known as lexical or word classes. POS tagging is called grammatical tagging as well as word-category disambiguation.

The process includes making up various words in some corpus in correspondence to certain particular part of speech mainly based on its context and its definition. Once performed, POS tagging is entirely done in the certain context of specific computational linguistics using different modes that commonly associated with certain discrete terms also including specific hidden parts of speech according to a collection of various disruptive tags. It should be noted that POS-tagging models commonly fall into two different groups including stochastic and rule-based group. In the following section, you will get to know how to use POS tagging in Python's NLTK. In order to begin, you have to import certain Python interpreter using function word tokenizer just before pos tagging.

```
import nltk
text = nltk . word _ tokenize ( " Dive into NLTK: Part – of - speech
tagging and POS Tagger " ) text [ ' Dive', 'into', 'NLTK', ':', 'Part
– of - speech', ' tagging', ' and ', ' POS ', ' Tagger ' ]
nltk . pos _ tag ( text ) [ ( 'Dive', 'JJ' ), ( 'into', 'IN' ), ( 'NLTK',
'NNP' ), (':', ':'), ( 'Part – of - speech', 'JJ' ), ( 'tagging', 'NN' ), (
'and', 'CC' ), ( 'POS', 'NNP' ), ( 'Tagger', 'NNP' ) ]
```

Fortunately, NLTK provides a huge amount of documentation for every tag that you will be using as a tag. The following step is to open NLTK help or some regular expression.

nltk . help . upenn _ tagset ('JJ')

nltk . help . upenn _ tagset ('IN')

nltk . help . upenn _ tagset ('NNP')

It should be noted that NLTK also provides a perfect method for POS tagging. In order to begin you will use function batch pos tag.

nltk . batch _pos _ tag ([[' this ', ' is ', ' batch ', ' tag ', ' test '], [' nltk ', ' is ', ' text ', 'analysis', ' tool ']]) [[(' this ', ' DT '), (' is ', ' VBZ '), (' batch ', 'NN'), (' tag ', ' NN '), (' test ', ' NN ')], [(' nltk ', ' NN '), (' is ', ' VBZ '), (' text ', ' JJ '), (' analysis ', ' NN '), (' tool ', ' NN ')]]

POS Tagging Model in NLTK

In this section, we will use NLTK already built-in POS tagging models. You can find all the needed code in maxent treebank pos tagging model. You will use standard treebank tagger. In this task, you will use that currently recommended part of NLTK POS tagging speech model in order to tag certain list of different tokens. Further, you will use the same model in order to tag sequence of tokens and to tag some given list of different sentences notably consisting of a collection of tokens.

POS _ TAGGER = ' taggers/ maxent _ treebank _pos _ tagger/english . pickle'

def pos _ tag (tokens)

from nltk . tag import pos _ tag # doctest: + SKIP

from nltk . tokenize import word _ tokenize # doctest: + SKIP

pos _ tag (word _ tokenize ("Lisa's big idea # doctest: +SKIP [(' Lisa ', ' NNP '), ("'s", ' PO S'), (' big ', ' JJ '), (' idea ', ' NN '), ('.', '.')]

tagger . tag (tokens)

```
def batch _pos _ tag ( sentences )
tagger = load ( _POS _ TAGGER )
return tagger . batch _ tag ( sentences )
```

The following step is to train your POS model. In order to do so, you will need to use the function maxent treebank tagging model set by default. Fortunately, NLTK provides this function in addition to other pos taggers like brill, CRF, and TNT alongside different interfaces in addition to Stanford pos tagging model, senna posttaggers and hunpos pos tagging models. I order to train your model you will use NLTK already built-in TnT tagger.

```
from __future __ import print _function
from math import log
from operator import itemgetter
from nltk . probability import FreqDist, ConditionalFreqDist from
nltk . tag . api import TaggerI
class TnT ( TaggerI )
```

It should be noted that it is possible to obtain that untrained POS tagger in order to create different tags for some unknown words using see init function. This function should be used only with certain sentence-delimited input since this tagger works the best when it is trained over some entirely sentence delimited input. On the other hand, it still produces proper results if the testing and training data are previously separated in all of the punctuation. It should be noted that your input for training data will be expected as a certain collection of sentences, and every sentence will be a certain list of tags, words, and tuples.

The following step is to train your test and train data. In order to do, you will use that already provided treebank data from NLTK's corpus. More specifically, you will use the initial 3000 treebank sentences as your train data while the rest of 914 sentences will be used as your test data. Further, you will train your POS tagger using train data in order to evaluate test data. The last step is to save your pos tagger model in pickle file. You are able to use is further any time you want.

Conclusion

Text analytics is all about converting some unstructured data into some meaningful data used for further analysis used commonly in order to measure various customer opinions, provide feedback or to entity modeling in order to support some mainly fact-based decision making. Text analytics techniques greatly help data analysis experts since these techniques are able of bridging that gap existing between marketer and analysts.

Advances in text analytics approach are common when it comes customer relationship and almost every other type of marketing industry. It helps marketers in order to organize as well as to classify that very confusing techniques while framing all the right questions. Further, text analytics helps marketer in order to find that successful meaning in almost every unstructured data notably helping them to develop some effective marketing strategies in better customer approach.

You already know that majority of data available to us is completely unstructured as well as text-heavy making it very challenging for data analysts to apply certain data wrangling in addition to visualization tools. However, using most common text analytics techniques those challenging tasks are things of past. If you are information professional, developer or if you are simply interested in creating and managing big data, this book will definitely help you on your journey.

Convolutional Neural Networks in Python

Introduction to Convolutional Neural Networks

By Anthony S. Williams

Table of Contents

Introduction

Deep learning is relatively newly introduced field of broader machine learning methods and research that has been introduced mainly in order to move machine learning research closer to artificial intelligence or simply AI. Deep learning in that matter also can be referred to as structured learning since deep learning is used in order to learn various tasks of deep learning neural networks that consist of multiple background or hidden layers.

Deep learning methods are all about learning large data representations or in other words learning neural network algorithms. It should be noted that deep learning could be performed in three varieties, supervised and unsupervised as well as semi-supervised deep learning.

Many deep learning interpretations are based on various information processing and different communication patterns. In fact, recent researches are devoted to defying types of relationship, which exist between different stimuli and corresponding neural network. Deep learning researchers are also devoted to creating some systems that would be able to learn various deep learning representations from large data collections or big data.

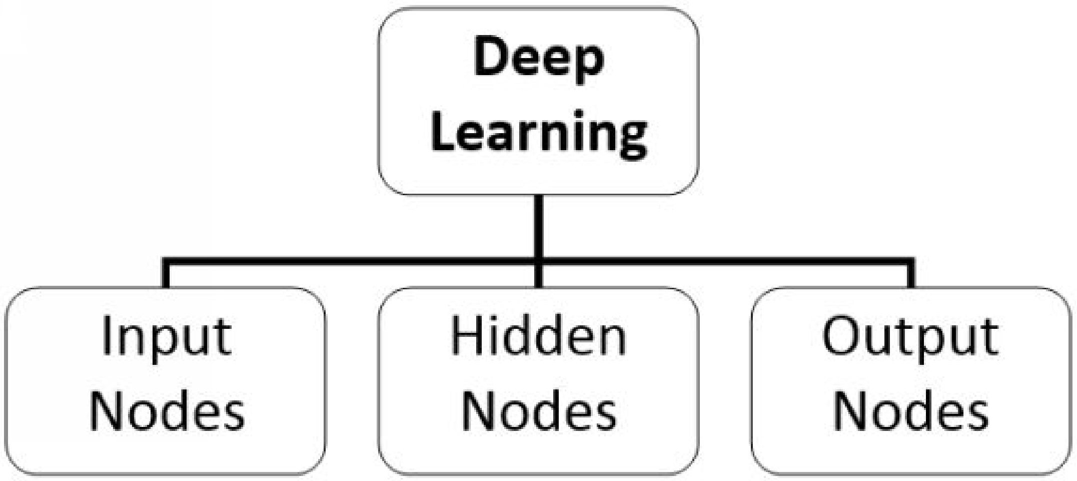
Various deep learning structures including deep neural networks like convolutional neural networks are widely used in different fields including speech recognition, computer vision, audio recognition, natural language processing, machine translation, social network filtering, bioinformatics and other. Like I mentioned before, deep learning is a part of a greater family of machine learning, and deep learning techniques include cascade containing many hidden and background layers of units that are represented in a nonlinear manner.

These units are used in deep learning methods in order to feature further transformation of an extraction. It should be noted that since these layers are represented in cascade pattern, every successive layer is, in fact, using the output from that previous layer as its own input. So here, you can see this cascade pattern in real meaning. Deep learning models might be completely supervised or on the other hand not supervised at all.

There are known as semi-supervised algorithms. Applications of deep learning models include classification that is completely supervised and pattern analysis that is entirely unsupervised. Deep learning techniques are also used as a certain method of learning a great number of levels of various features as well as representation of large data collections or big data.

Hierarchical deep learning representation is a widely used method since it uses features from higher levels to generate those lower level data features and representations. In other words in hierarchical data concept represent a great number of data representations which correspond to various abstract levels.

To summarize deep learning, we can simply say that deep learning methods use various layer units and both supervised and unsupervised learning of various data feature representations are shown in each of these layers. Layers further form a hierarchical structure, which is in the form from lower to higher level features. Layers that are formed in a nonlinear manner and their composition mainly depend on which problem is going to be solved.



There are various layers, which have been used in order to solve deep learning problems. This book is focused on convolutional layers that are used in different fields including computer vision, audio recognition, various image effects and other. As soon as we get familiar with deep learning and neural networks in general, we are going to see one of the most typical applications of a convolutional neural network, and there will be more word of it in next chapters.

Deep learning methods lie under an assumption that hidden level representations in gathered data are in fact, generated by the relationship and interaction of various features of these layers. Also further, there is an assumption that these certain layers and their features, in fact, correspond to multiple levels of composition and certain levels of abstraction. It should be noted that a different number of layers as well layer sizes might provide a wide range of different structures and abstractions.

Deep learning:

- *Using large datasets*
- *Learning complex problems*
- *Automatically learning data features*

Deep learning techniques actually exploit a particular idea of the hierarchical concept of various factors and features in circumstances where layers form higher levels are generated from those from lower levels. In other words, higher level features are learned from lower layer features.

Deep learning models are commonly generated with a layer-by-layer algorithm where deep learning methods help in order to disentangle various abstractions and compositions in order to pick out those useful and relevant features that will eventually improve overall performance.

Definition of Deep Neural Networks

Deep neural networks are mainly biologically inspired outstanding paradigm that enables any computer to learn a broad range of useful and relevant information just from observing different data. On the other hand, deep learning models are in fact various deep learning techniques when it comes to the data contained within deep neural networks.

Deep learning and neural networks currently provide the best examples of how great data can be. They also provide the best possible solutions to various problems in different fields including natural language processing, speech and image recognition and other.

This book is mainly focused on a great part of neural networks, convolutional neural networks that are beyond useful in audio and image recognition, so further great examples of convolutional network applications will be discussed.

Further chapters will also teach you what is really behind convolutional neural networks and how to perform various image and audio recognition using convolutional networks in Python. However, we will cut to the chase in the next chapter.

Neural networks are in fact a various collections of algorithms and deep learning models that are loosely modeled after the structure of the human brain. Neural networks are mainly designed in order to recognize different patterns of big data. In order to do neural networks interpret different sensory data through a type of machine learning perception as well as through clustering and labeling raw inputs.

Deep Neural Networks:

- *Inputs*

- *Weights*
- *Function of net input*
- *Activation function*
- *Output*

The patterns that are recognized by various deep neural networks are in the numerical form contained in different bias vectors. Into these bias vectors, all data from the real world like sound, text, time series and images at some point during deep learning will be translated. Neural networks are designed in order to help us classify and cluster a large collections of data. In other words, think of neural networks as classification and clustering layer that is placed on top of data that will be managed and stored.

Neural networks are also designed in order to help us to group various data that is unlabeled according to certain similarities that exist within different data collections. Data example inputs are also similar on some occasions, and neural networks classify that data when there is a labeled data collection available for data training process. It also can be said that neural networks extract various data features which are fed to different algorithms for further classification and clustering. Think of neural networks as various components of the larger family of machine learning models and applications. Neural networks involve different algorithms designed for data regression, data classification and reinforcement deep learning.

Deep Neural Networks		
Reinforcement Learning	Data Classification	Regression

In order to use neural networks properly, you first have to ask yourself a couple of questions since various neural networks are used in order to obtain a different solutions. Therefore, depending on what outcome is relevant to you, you will use different neural networks to come to a useful solution. Deep learning techniques are all about various inputs to different outputs which outputs, which are in fact a universal correlation. Deep learning algorithms provide us these universal correlations between inputs and outputs.

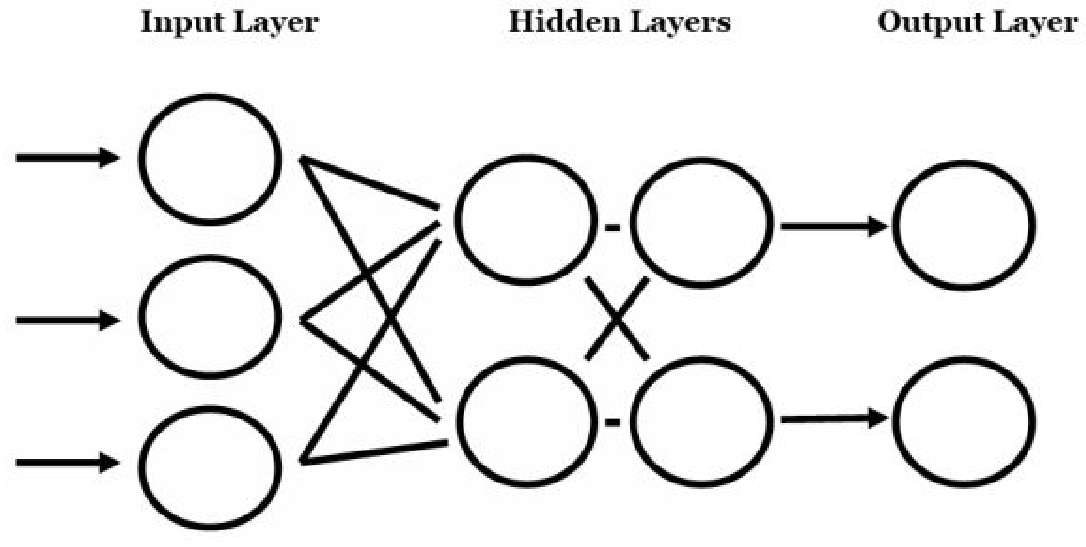
Deep learning techniques also are able to provide us approximate of the function between any possible input and any potential output. We just have to assume that inputs and outputs are related via certain causation or correlation. The deep learning process can find the right solution to equation including various inputs and outputs and these processes also find the correct manner of different transformation of inputs into outputs.

Deep Learning Equation:

$f(x) = y$

x is any possible input

y is any possible output



Deep Learning Classification and Clustering

All deep learning classification is processed upon individual labeled data collections. In other words, classification is done when human transfer their relevant knowledge to those data collections in order to neural networks learn various correlation that is present in these data and labels. This deep learning process is known as supervised deep learning technique. In the following section you will learn how to create a convolutional neural network using Python. The first step is to import your sequential model.

```
from keras.models import Sequential
from keras.layers import Dense
import numpy
numpy.random.seed(7)
pima-indians-diabetes.csv
dataset = numpy.loadtxt("pima-indians-diabetes.csv",
delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8] 1 2 3 4 5
```

The following step is to define your model.

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X, Y, epochs=150, batch_size=10)
scores = model.evaluate(X, Y) print("\n%s: %.2f%%" %
(model.metrics_names[1], scores[1]*100))
```

Supervised deep learning model can identify various people in images as well as detect faces on these images. Supervised deep learning also can provide us with detection of various facial expressions like joyful and angry. These techniques are also able to identify various objects that are contained in the images like different lane markers.

Using supervised deep learning, you also can recognize different gestures in videos detect voices, identify different speakers. In addition, you can transcribe different speech into text format, recognize different sentiments in voices and other. In order to cluster and classify a convolutional neural network in Python, you will start by importing data. The source code is represented above.

```
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
from sklearn.cluster import KMeans
style.use('ggplot')
X = np.array([[1, 2], [1.5, 1.8], [5, 8], [8, 8], [1, 0.6], [9, 11]])
plt.scatter(X[:, 0],X[:, 1], s=150, linewidths = 5, zorder = 10)
plt.show()
```

Supervised deep learning process also can classify various texts as spam like in emails or classify texts as fraudulent like in insurance claims. Supervised deep learning is also used in order to recognize sentiment in various text format data like customer feedback. In other words, deep learning processes are able to generate any outcome that is useful and relevant to you and which is in certain correlation to various data you provided. Any of labels that are created by a human, in fact, can be used in order to train different neural networks.

On the other hand, deep learning clustering is grouping based on certain similarities. It should be noted that deep learning methods do not require specific labels in order to detect various data similarities. Deep learning when these particular labels are not included is called as unsupervised deep learning. In fact, the majority of worldwide data is, in fact, unlabeled data. In the followings section you will see hot to integrate clustering using Python using the k-means model. In order to start you have to import required packages.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.cluster import KMeans
import utilities
```

The next step is to input data and further define the number of all clusters.

```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
TRAIN_DIR = ‘X:/Kaggle_Data/dogs_vs_cats/train/train’
TEST_DIR = ‘X:/Kaggle_Data/dogs_vs_cats/test/test’
IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = ‘dogsvscats-{}-{}.model’.format(LR, ‘2conv-basic’)
```

You will split put the labels and convert them to an array.

```
def label_img(img): w
ord_label = img.split('.')[-3]
word_label == 'cat': return [1,0]
elif word_label == 'dog': return [0,1]
```

You will build another function in order to fully process the training images.

```
def create_train_data():
training_data = []
for img in tqdm(os.listdir(TRAIN_DIR)):
label = label_img(img)
path = os.path.join(TRAIN_DIR,img)
img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
training_data.append([np.array(img),np.array(label)])
shuffle(training_data)
np.save('train_data.npy', training_data)
return training_data
```

You will save and return the array data.

```
def process_test_data():
testing_data = []
path = os.path.join(TEST_DIR,img)
img_num = img.split('.')[0]
img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
testing_data.append([np.array(img), img_num])
```

```
shuffle(testing_data)
np.save(‘test_data.npy’, testing_data)
return testing_data
```

You will run the training.

```
train_data = create_train_data()
```

The next step is to define your neural network.

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout,
fully_connected
from tflearn.layers.estimator import regression
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1],
name=‘input’)
convnet = conv_2d(convnet, 32, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = fully_connected(convnet, 1024, activation=‘relu’)
convnet = dropout(convnet, 0.8)
convnet = fully_connected(convnet, 2, activation=‘softmax’)
convnet = regression(convnet, optimizer=‘adam’,
learning_rate=LR, loss=‘categorical_crossentropy’,
name=‘targets’)
model = tflearn.DNN(convnet, tensorboard_dir=‘log’)
model.load(MODEL_NAME)
print(‘model loaded!’)
```

The following step is to split testing and training data and create your data arrays.

```
train = train_data[:-500]

test = train_data[-500:]

X = np.array([i[0] for i in
train]).reshape(-1,IMG_SIZE,IMG_SIZE,1) Y = [i[1] for i in train]

test_x = np.array([i[0] for i in
test]).reshape(-1,IMG_SIZE,IMG_SIZE,1) test_y = [i[1] for i in
test

model.fit({‘input’: X}, {‘targets’: Y}, n_epoch=3, validation_set=
({‘input’: test_x}, {‘targets’: test_y}),

snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

One of the principal laws of machine learning states that the more data that is operable by the deep learning algorithm, the greater accuracy will be. Therefore, in other words, we can say that unsupervised deep learning has a great potential of producing highly effective and accurate algorithms.

Data Classification and Clustering		
Criteria	Classification	Clustering
Already gathered knowledge of classes	Yes	No
When it is used	Classify provided sample into already known classes	Suggest collections which are based on specific

		patterns within data
Models	Bayesian classifiers and decision trees	Expectation maximization and K-means
Data requirements	Labeled samples from a collection various cases	Samples that are unlabeled

Deep learning unsupervised clustering processes in fact search through various data including documents, multiple images and other. Further deep learning process includes comparing these gathered data to some similar items.

Unsupervised deep learning processes also include a method for anomaly detection. The greatest flip side of identifying similarities is that detecting similarities is, in fact, detecting various anomalies or any other unusual data behavior. In a high number of cases, this unusual data behavior really correlates greatly with that thing that you intended first to detect and then prevent, such as various spam email and fraud.

Besides deep learning classification and clustering, deep learning processes also can establish various correlations between different data such as pixels contained in an image as well as the name of a certain person who is also on that image. This is referred to as static prediction. Deep learning techniques are also used to establish various correlations when there is enough of exposure of data gathered. These correlations are between current and future events. This means that this future event prediction is within the label in main sense.

Deep learning techniques not necessarily care about time principle. These techniques are not necessary in the case if something happened or it will happen at some time eventually. Provided a time series, deep learning techniques are able to read various strings of numerical data and predict which number is most likely to occur in next sequence. Deep learning predictive analytics is commonly used for hardware breakdowns including manufacturing, transport and data centers. Predictive analytics is also used when it comes to the various health breakdowns including heart attack that is based on person's vital stats when data from health wearables is gathered.

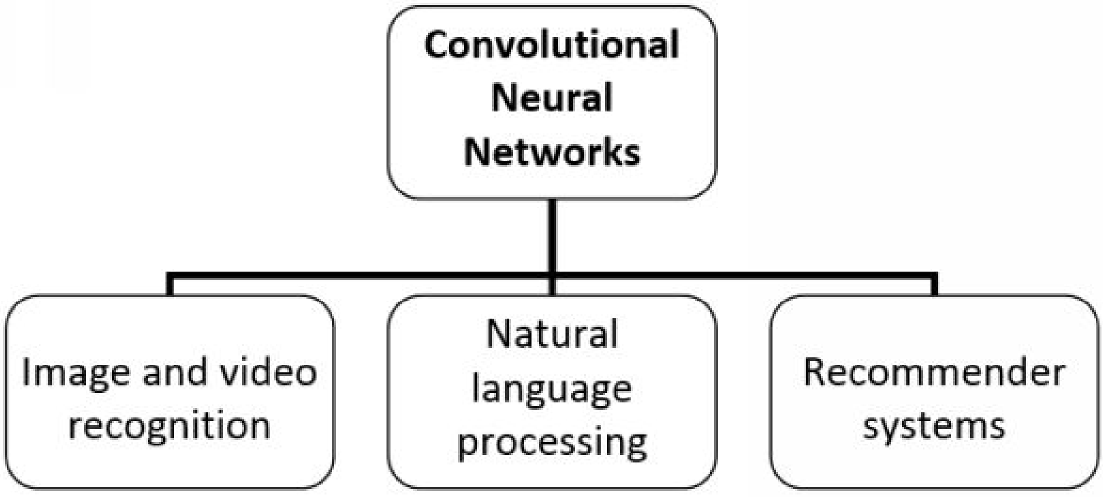
Supervised and Unsupervised Deep Learning	
Supervised Classification	Unsupervised Clustering
Number of classes if known	Number of classes is unknown
Based on a training set	No prior knowledge
Used to classify future observations	Used to explore and understand data

Chapter 1 Introduction to Convolutional Neural Networks

Convolutional neural networks are part of a broader family of feed-forward networks that are always composed of multiple convolutional layers containing completely connected layers. In order to understand better what convolutional neural networks are, we have to discuss more about feed-forward network family.

Types, of Neural Networks:

- *Single neural network- linear and logistic regression*
- *Feed-forward neural network- no cycles, nonlinear regression, and classification*
- *Symmetric RMB neural network- unsupervised, trained in order to maximize likelihood of input data*


Convolutional Neural Networks
Image and video recognition
Natural language processing
Recommender systems

Feed Forward Networks

Feed-forward networks are artificial neural networks where the certain structure lying between neural networks is not in the form of a cycle. This, in fact, is the main difference between feed-forward and recurrent neural networks. The feed-forward neural networks are firstly designed since this neural structure is the simplest type of deep neural networks in general.

In this particular network, the various data information is able to move only in a single direction that is forward. The information is transferred from the input through hidden nods and finally to the output nodes. There is no any cycle formed in this particular neural structure. The simplest kind of feed-forward network is a single layer network that consists of only one layer of outputs, On the other hand, the inputs are fed to the neural outputs via certain weight. Therefore, a single layer neural network is a basic type of feed-forward networks.

In each of single layers nodes, the sum of all products and all weights are calculated. In the case when estimated value is above zero in most cases, neuron further fires and takes value one in most common cases since one is activating value. On the other hand, if the total value of calculated inputs and weights is not above a certain threshold, that is mostly zero, and then neuron will take value minus one that is deactivating value in most cases. These neurons are often referred to as artificial neurons as well as linear threshold elements.

Feed-forward network:

- *Train a neural network with training set*
- *Test a neural network using the test set*
- *Classify data*
- *Evaluate outcome*

A certain perception can be created if you use any of these values for both deactivated and activated state as long as the value of the threshold lays between the two. These perceptions also can be trained by very simple deep learning model, which is commonly called as the delta rule. The delta rule, in fact, can calculate various errors that are between these calculated outputs and other sample output information. The delta rule further uses this calculation in order to create an adjustment to the calculated weights by implementing a certain form of gradient descent.

Single-layer perceptions are just capable of deep learning some linearly independent patterns. Even though single-layer elements are quite limited regarding its computational power, it is proven that networks that are from parallel threshold elements are able to approximate various continuous function structure from various compact intervals consisted of real numbers into one and minus one interval.

On the other hand, multi-layer neural networks are able to compute a continuous output rather than a step function. Computing these continuous outputs is called a logistic function. With some choices, single-layer networks are identical to these logistic regression structures that are commonly used in statistical modeling. The sigmoid function is another very common name of the logistic function. The sigmoid function, in fact, is consisted of a continuous derivative that allows being used as back-propagation.

Multi-layer perception feed-forward neural class consists of a great number of layers of computational elements that are usually interconnected in a certain feed-forward structure. Each neuron that is within a single layer has certain directed connections to other neurons that are within the subsequent layer. In various applications, the elements of this particular neural networks apply previously mentioned a sigmoid function in the form of an activation function.

An example code for feed-forward backpropagation of single convolutional neural network.

```
class Neuron(object)
def forward(self, inputs)
cell_body_sum = np.sum(inputs * self.weights) + self.bias
```

```
firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum))
return firing_rate
```

There is the universal approximation theorem related to neural networks, which state that each continuous function, which maps various intervals, consisted of real numbers to some certain output intervals consisted of real numbers, in fact, can be approximated closely by these multi-layer perceptions that consist of a single hidden layer. These results hold for a great number of functions.

Multi-layer neural networks also use a great variety of various deep learning techniques and the most popular definitely is back-propagation.

Using the back-propagation methods, the certain output values are being compared with the different correct answers in order to compute the certain values or already defined error-function. By different techniques, the error is further fed back through that particular neural network. Using gathered information, the model further is able to adjust weights of every connection to reduce a certain value of that error function by some other smaller amount.

Multi-layer Feed-forward Network:

- *Contains one or more hidden layers*
- *Computation nodes are called hidden nodes or hidden neurons*
- *Function of hidden nodes is to inter-mean the external input and network output*

When this process is repeated for a sufficient number of subsequent training cycles, that certain neural networks will commonly converge to some other state where that error of the previous calculations is very small. In this particular scenario.it can be said that the neural network has learned a specific target function. In order to adjust weights correctly apply a general technique for nonlinear optimization. This technique is generally referred to as gradient descent.

In order to achieve this, the network will calculate the derivative of the gathered error function concerning the provided network weights. The further neural network will change these weights like those that specific error decreased. For this reason, back-propagation is only applicable on neural networks that have differentiable functions for activating.

There is generally a problem when it comes to the teaching a specific neural network how to perform well and execute relevant solutions when it comes to the using samples which are not previously used as samples for data training. In this case scenario, some additional methods and techniques may be required to teach a neural network to perform great. This is most important in cases where there is an only limited value of numbers as well as a limited number of samples that are available. The risk is great since network can over-fit that training data which means it fails when it comes to the capturing the true process of generating relevant data.

Deep learning computational theory is greatly concerned with these training classifiers when it comes to the very limited amount of data, which is available. It is commonly stated that neural networks will perform well when it comes to the generalizing examples that are not within the training data collection. Another common problem of the back-propagation structures is, in fact, the speed of data convergence as well as the possibility of getting to a local minimum of that error function.

However, currently, there are great methods, which make this process regarding the multi-layer perception the great tool of various choices for different deep learning tasks. You also have an ability to use a multiple series of not dependable neural networks that are moderated by some similar and intermediary behavior. In other words, their neural networks are capable of performing independently as well as handling large tasks, and further these results can be combined into one solution.

Our main goal is to use different neural network nets for them to arrive at a certain point where there are at least possible error happened. In addition, this task should be done as fast as possible with minimum risk. Think of it like running a very important race, and the time is the key towards a good performance and expected solution. The deep learning process can be perfectly described as running a race since at some point you run on the same part of the track for multiple times, and you are going in a circle. Just like runners, we also have to engage in some repetitive acts for multiple times in order to arrive at the certain finish point. At the starting line, all weights are completely initialized, and finish line is, in fact, the true state of various parameters that are capable of producing accurate predictions and classifications.

Each step for any neural network involves some guessing, various error measurements as well as a slight update of its weights. Coefficients and their certain adjustments are also required. Weight sets either in an initial state, and their end state is referred to as neural model since it is, in fact, a certain step towards modeling various data connections to different labels that are ground-truth in order to have a look at Data's architecture. Models generally start out poorly, but at the end, they are less bad since the models are changing over time since neural networks regularly update models' parameters. Neural networks are in fact created in ignorance, and so they do not know which particular biases and weights will work in order to translate the input in order to make the all correct guesses. Everything really starts with the certain guess and further we try to make better sequentially guesses since neural networks learn from their previous mistakes.

So think of neural networks as a small enactment of other scientific methods like delivering hypothesis, which requires constant and repeated trying just as neural networks perform. In feed-forward neural networks, the first input will be entering the network. In the end, certain coefficients, as well as weights, will be placed. A collection of inputs is also at the end.

Input times weight is equal to certain guess

Certain weighted inputs result in certain guesses on what that certain input is about. The neural network further takes other guesses that are compared to ground-truth about that particular data.

Ground truth minus guess is equal to error

These definitions above account for three fundamental functions of all neural networks and that are scoring different outputs, calculation various loose and applying an update to the algorithm and these fundamental function begin this three-step process in repeated manner. In other words, neural networks are corrective loops while rewarding weights which supports neural networks' correct guesses. On the other hand, they are punishing those weights that are leading to errors.

Error times contribution of weights to error is equal to adjustment

In the following section you will see how to incorporate a convolutional neural network in Python using different weights. In order to start you have to define your function.

```
def conv_forward(X, W, b, stride=1, padding=1): pass
```

Compute the bias gradient.

```
db = np.sum(dout, axis=(0, 2, 3)) db = db.reshape(n_filter, -1)
dout_reshaped = dout.transpose(1, 2, 3, 0).reshape(n_filter, -1)
```

```
dW = dout_reshaped @ X_col.T
dW = dW.reshape(W.shape)
W_reshape = W.reshape(n_filter, -1)
dX_col = W_reshape.T @ dout_reshaped
def conv_forward(X, W, b, stride=1, padding=1):
cache = W, b, stride, padding
n_filters, d_filter, h_filter, w_filter = W.shape
n_x, d_x, h_x, w_x = X.shape
h_out = (h_x - h_filter + 2 * padding) / stride + 1
w_out = (w_x - w_filter + 2 * padding) / stride +
h_out.is_integer()
w_out.is_integer()
h_out, w_out = int(h_out), int(w_out)
X_col = im2col_indices(X, h_filter, w_filter, padding=padding,
stride=stride)
W_col = W.reshape(n_filters, -1)
out = W_col @ X_col + b
out = out.reshape(n_filters, h_out, w_out, n_x)
out = out.transpose(3, 0, 1, 2)
cache = (X, W, b, stride, padding, X_col)
return out, cache
def conv_backward(dout, cache)
n_filter, d_filter, h_filter, w_filter = W.shape
db = np.sum(dout, axis=(0, 2, 3))
db = db.reshape(n_filter, -1)
dout_reshaped = dout.transpose(1, 2, 3, 0).reshape(n_filter, -1)
dW = dout_reshaped @ X_col.T
dW = dW.reshape(W.shape)
W_reshape = W.reshape(n_filter, -1)
```

```
dX_col = W_reshape.T @ dout_reshaped
dX = col2im_indices(dX_col, X.shape, h_filter, w_filter,
padding=padding, stride=stride)
return dX, dW, db
```

Architecture of Convolutional Neural Networks

The deep learning state of the art in 2011 alternated convolutional deep networks as well as maximum-pooling networks that are consisted of multiple commonly entirely connected layers, which are followed by classification layer. Convolutional neural networks are used in unsupervised deep learning. These non-supervised deep learning techniques were, in fact, the first artificial pattern used for recognizing in order to achieve great human-competitive performance in various tasks.

Convolutional Neural Networks:

- *Widely used to image data*
- *Translation invariance*
- *Local receptive fields*
- *Weight sharing*
- *All the elements in the convolutional layer will detect the same data patterns, but at different locations within the input image*

Convolutional neural networks are created in order to process various data, which came in the form of numerous arrays. For instance, a colorful image that is composed of three two-dimensional arrays that are also containing various pixel intensities in those three channels of color. In fact, many data learning models, are in this form that contains multiple arrays. These multiple arrays commonly include one dimensional for sequences and signal, two-dimensional for audio and image spectrograms and three-dimensional for volumetric pictures and videos.

Behind convolutional neural networks, there are four fundamental ideas, and all of them take a certain advantage of natural signals and their properties. These natural signal properties include shared weights, the use of multiple layers, local connection and pooling.

Typical architecture of convolutional neural networks is in the form of multiple stages. The initial stages consist of two different types of layers, and those are pooling and convolutional layers. Elements that are within convolutional neural networks are arranged in certain feature map, and within that map, each element is connected to some local patches within that same feature map that also contains the previous convolutional layers using a collection of certain weights that are referred to as a filter bank.

Convolutional Neural Networks:

- *Convolutional layers*
- *Max-pooling layers*

The result of these local weights is further transferred through in a nonlinear manner like a ReLU. It should be noted that each element within feature map shares completely same filter bank. On the other hand, different maps of a feature in an each layer use other filter banks. Therefore, the structure of convolutional neural networks is in fact twofold.

In convolutional neural networks, various array data like images and local collections of values are commonly highly correlated in order to form various local motifs that will be easily detected. The second reason for a two-fold structure of convolutional neural networks is the statistics of various images as well as other signals that are in fact invariant to a certain location. It can be put this way as well, if the certain motif can appear in some part of the certain image, it as well can appear anywhere on that same image.

The main idea of these elements is that they are using various locations but at the same time, they share weights as well as detect the same pattern that is in different parts of that certain array. It also can be put in mathematical meaning, so image filtering using convolutional layers can be performed by various feature maps, which are referred to as a discrete convolution.

The main role of convolutional neural networks is detecting some local conjunctions of various features that are gathered from the previous layers. On the other hand, the role of other layers like pooling layers is to combine features that are semantically similar to a single one. Since the relative positions of various features may vary, adequate detecting the various motifs will be done by process of coarse-graining the every single position of every feature.

A common pooling element, in fact, is able to compute the maximum of various local patches of elements in a single feature map as well as in multiple feature maps. It should be noted that neighboring elements commonly take inputs from different patches, which are shifted, by multiple columns and rows. Therefore, they are in fact reducing various dimensions of that certain representation as well as create certain invariance to smaller distortions and shifts.

Multiple convolutional stages including both pooling and nonlinearity are stacked, and they are further followed by multiple convolutional layers that are entirely connected. In addition, it should be noted that back-propagation gradients throughput convolutional neural networks are simple as throughout the regular deep learning networks and all weights that are in filter bank are easily trained.

Convolutional neural networks, in fact, exploit various properties and many signals are in fact compositional hierarchies, and higher-level features are gathered by generating lower level elements, which are exactly what hierarchy structure represents. For instance in images, edges and their local combination are those structures that form various motifs, further these motifs assemble into some parts, and finally, these parts form different objects.

Structures similar to these also exist in speech as well as text from sounds to syllables, words, sentences, phones, and phonemes. In fact, it is pooling that allows different representations to very when elements in scenarios when different elements from the previous convolutional layers also vary in appearance and position.

Both pooling and convolutional layers are inspired by some classic notions or very simple and very complex cells regarding visual neuroscience. The overall convolutional neural network structure is, in fact, a hierarchy in these visual contexts. When convolutional neural networks structure and algorithm are shown the same image, high-level elements and theirs activations explain almost a half of the various variance in random collections. It also should be noted that convolutional neural networks have some of their roots in the scientific field of recognition since the neurocognition structures are very similar to those of convolutional neural networks, but neurocognition structures do not feature end-to-end deep supervised learning models like back-propagation. A very primitive convolutional neural network one-dimensional also called as time-delay neural structure was commonly used in order to recognize various phonemes in different simple words.

Applications of convolutional neural networks are great since the very beginning dating back to 1990s starting with these one-dimensional time-delay convolutional neural networks especially used in document reading and speech recognition. These early document readings done by convolutional neural networks were trained jointly using some probabilistic model, which implemented different language constraints.

In the late 1990s, this particular system was used in reading over ten percent of all cheques in the USA, so convolutional neural networks were of great importance at that time. In the following years, various convolutional neural networks-based handwriting and optical character recognition systems were developed by a giant in the industry Microsoft. Convolutional neural networks were also used in experimenting with various object detection systems in natural images including hands and faces as well as for systems used in face recognition.

Convolutional Neural Networks:

- *Input*
- *Feature maps and convolutions*
- *Feature maps and sub-sampling*
- *Fully connected layers*
- *Output*

Chapter 2 Image Understanding

Convolutional neural networks have been widely used and applied with great success in order to detect recognition and segmentation of various objects and areas in images. These tasks are those tasks where labeled data is relatively abundant like traffic sign recognition as well as the segmentation of various biological images especially for connectomics and detection of different faces, human bodies, pedestrians in natural images.

The major success of convolutional neural networks is in face recognition systems. Numerous important images, in fact, can be labeled at some pixel level that will have various applications in technology and that application include autonomous robots as well as self-driving cars. Some of the major companies including NVIDIA and Mobileyes use convolutional neural networks based technique for their upcoming vision systems that are currently used in cars. Other convolutional neural networks applications include speech recognition and natural language processing.

Image Recognition using Convolutional Neural Network

- *Image classification*
- *Single-object localization*
- *Object detection*

Despite this great success in various scientific fields, convolutional neural networks have been greatly deserted by mainstream computer-vision as well as machine-learning worldwide communities until 2012 and the ImageNet competition that showed what great opportunities convolutional neural networks offer.

In cases when deep convolutional networks are applied to a certain data collection that contains about million images from all over the web which contain more than thousand different data classes, they can achieve outstanding results, and the error rates are similar to those of the best other machine-learning approaches.

This convolutional neural networks success mainly came from well used GPUs and ReLUs as well as newly adopted regularization technique which is referred to as dropout technique. These techniques are used in order to generate greater training data examples by deforming those that are already gathered. This convolutional neural network success really has brought an ultimate revolution in computer vision systems and techniques.

Today, convolutional neural networks are commonly used for almost every detection and recognition task since they approach almost a human performance on certain tasks. Some of the recent uses of convolutional neural networks include combining these with recurrent neural networks in order to generate various image captions.

Some of the recent convolutional neural network structures contain from ten to twenty layers of ReLU and millions of various weights as well as billions of different connections between elements. Recent progress in hardware, algorithm parallelization, progress in software greatly has reduced data training time just too few hours which is a great success since at the very beginning a time needed for data training have taken several weeks.

The great performance of convolutional neural networks has caused some great changes in some of the major companies such as Facebook, Google, Adobe, Twitter, Microsoft, IB and Yahoo to initiate various research as well as various development projects in order to deploy convolutional neural networks based image recognition and detection systems. So, all these major companies use convolutional neural networks for various detection and image recognition tasks.

Convolutional neural networks are also very easily amenable in order to efficiently implement various hardware field gate and chips arrays. Therefore major companies such as Intel, NVIDIA, Mobileye, Samsung, and Qualcomm are currently developing convolutional neural networks based chips in order to enable real-time vision different applications in robots, cameras, smartphone and self-driving cars.

Computer Vision Using Convolutional Neural Networks

Beyond our imaginations is most certainly the real power of artificial intelligence and deep neural networks that offer great opportunities and some of them are even imaginable. In some countries of the world, it is well known that robots already have reached a certain testing phase. On the other hand, worldwide governments, as well as major companies, are spending a great amount of money in order to develop ultra-intelligence artificial creature. These recent launches of intelligent robots caught attention worldwide, and that is not surprising.

Image Recognition in Python using a convolutional neural network.

```
from SimpleCV import Camera
Initialize the camera cam = Camera()
Loop to continuously get images while True
Get Image
from camera img = cam.getImage()
Make image black and white
img = img.binarize()
Draw the text “Hello World” on image img.drawText(“Hello
World!”)
Show the image img.show()
```

Computer Vision:

- *Signal processing*

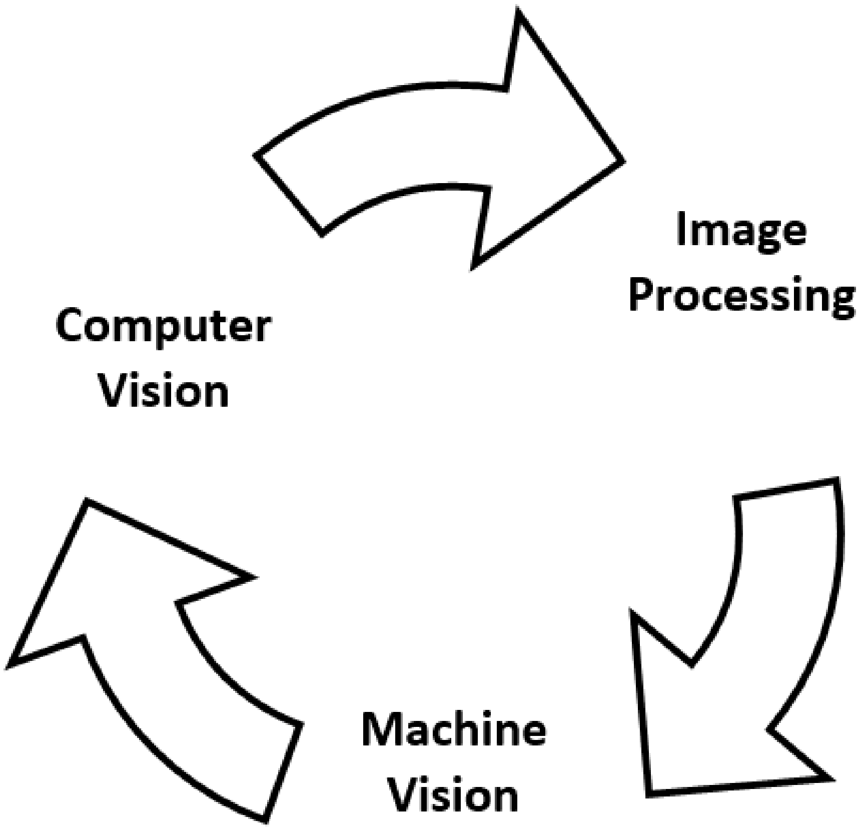
- *Mathematics*
- *Artificial intelligence*
- *Pattern recognition*
- *Physics*
- *Image processing*

When it comes to the computer vision based research those started in the middle of 1950s, and these tasks were greatly difficult, but since the introduction of deep learning techniques and neural networks, computer vision tasks are easier to solve than ever before. Application of deep learning techniques and models really have taken computer vision technology to a whole new level, and this new level is more sophisticated incorporated into various things such as self-driving cars that are are our future..

Convolutional neural network based systems are widely used in computer vision, and they are most certainly the crux of deep learning method applications particularly when it comes to the computer vision systems.

When it comes to the discussing approach used in computer vision, first some challenges that might be encountered should be discussed. The main goal of computer vision as the name suggest already is to imitate various functionality of human brain and eye components that are responsible for our sense of sights. Doing various actions like recognizing various animals, describing the certain view, differentiating between various visible objects are easily done by humans.

The main aim of computer vision is to make these actions possible by computers in human alike performance. In reality, it really took more decade of dedicated research to finally discover as well as impart the ability to recognize and detect various objects to a computer having reasonable accuracy and well performance.



This field of computer vision has witnessed various advancements, especially in a few recent years. Convolutional neural networks are most certainly one of the greatest advancement. Currently, deep convolutional neural networks are really the crux of greatly sophisticated computer vision systems and applications including self-driving cars, facial security features, auto-tagging friends on various social media platforms, different gesture recognition, plate number recognition and many other.

One of the fundamental and most basic applications when it comes to the computer vision is object detection. Further computer visions developments are in fact achieved by making enhancements on top of the fundamental object detection. Think of it as the same things occurring in real life, since every time you open your eyes you unconsciously detect various objects that are around you. Since we do this object detection unconsciously, we do not appreciate various fundamental challenges that are involved when someone tries to design a system that is similar to human eye.

There are many challenges when it comes to the computer vision, and one of them is most certainly great variations in viewpoint. In other words, different objects may have various positions as well as angles in a single image that is, in fact, depending on the relative position both of the observer and object. Another challenge related to the variation in viewpoint is different positions.

For instance, if you look at the same object your point of view may be different, and you see that same object from various angles. Moreover, it is easy to you since you know that even if it appears that the object is different, you know that you are looking at the same object. This is something that is hard to teach to a computer or machines.

Another computer vision challenge is a difference in illumination. Various images can have various light conditions, and we can easily detect the object in the images regardless of illumination. However, teaching to a computer this aspect is another great challenge. Other commonly used computer vision systems great challenges are hidden parts of images. It is not necessary for images to be completed and some parts of the image smaller or larger can appear hidden. This hidden part of the images makes the recognition and detection a greater task.

Background clutter is another challenge for computer vision since it may happen that some images blend completely into the image background. It is a great challenge to learn to a computer how to recognize and detect an object on the images that might blend in with the background.

These computer vision challenges really speak loud, so you can appreciate this great complexity of the various tasks which your brain and eyes do with utter ease. Recognition, detection and breaking up various challenged and solving them separately are possible today in computer vision using convolutional neural networks.

However, it should be noted we are far away from the system that would be able to get close to human eye performance. In fact, the brilliance of human body was the main reason for major researchers to try breaking the computer vision enigma and their fundamental approach is based on the visual mechanics of human eye.

Computer Vision		
High Level	Medium Level	Low Level
Vehicle detection and lane departure	Image segmentation and connected components	Edge detection and homography estimation
Pedestrian detection and face detection	Harris corner detection	Histogram and convolution kernels
Stereo vision and object tracking	Lens distortion detection	Morphological operators

Some of the earliest computer vision tasks were done by Weisel and Hubel in 1959 which is known as the famous cat experiment. What this cat experimented was, in fact, the first computer vision study that emphasized how great importance of various edge detection system when it comes to the solving various computer vision problems. Weisel and Hubel were also awarded the Nobel Prize for their dedicated work in this particular field. Traditional techniques that have been used in computer vision systems are replaced by sophisticated convolutional neural network based systems. These techniques that are not deep learning were commonly used, and they are still used in order to solve simple computer vision tasks. When data collection became large, and the task is more complex, there is no any substitute for convolutional neural networks.

Some of the simple approaches for enhancing the computer vision include K-nearest neighbors or KNN techniques where every image is matched with all other images contained in training data. Using this computer vision enhancing technique the top K containing minimum distance will be selected. The majority of class that is selected will be predicted as a certain output class of that particular image.

Computer Vision:

- *Noise removal and image sharpening*
- *Object recognition segmentation*
- *Scene understanding and autonomous navigation*

In addition, various distance metrics will be used like sum of the absolute distance that is denoted as L1 and L2 or a sum of squares and other metrics. A common drawback of KNN's computer vision enhancing technique include some illumination challenges as well as object orientation. For instance, if we take an object that has the same orientation and illumination, the object still may lie in various locations of that particular image.

Another traditional computer enhancing technique is linear classification where a parametric approach is used. In this technique, every pixel value will be considered as a certain value.

In other words, this technique can be explained as the weighted sum of various pixel values that have dimensions of certain weights matrix that is depending on the possible outcomes number. This technique encounters same computer vision challenges as the previous technique.

Fundamentals of Neural Networks

Some of the fundamentals of neural networks including convolutional neural networks include activation functions. There are different activation functions that can be used. One very common activation function is a Sigmoid function that is widely used in logistic regression, and since this activation function has some issues, it is never used in convolutional neural networks. Some of the issues regarding the Sigmoid function are that saturated neurons likely kill the gradient. Another issue of this activation function is that outputs are not centered at zero.

Other activation functions included tanh activation and rectified linear unit of ReLU function. Tanh activation function is one of the simplest activation functions presented as hyperbolic tangent. This activation function is in most cases preferred over the Sigmoid function since output are always in the range one and minus one. On the other hand, just like the Sigmoid function, it also kills the gradient which makes this activation function not a great choice.

Activation function that is a great choice is ReLu or rectified linear unit. This function is most commonly used in convolutional neural networks since it offers some great advantages. Some of the ReLU activation function advantages include gradient that will not saturate in any positive area. ReLu function is also very efficient computationally, and only a simple threshold is required. ReLU activation functions are empirically found to converge in a faster manner than tanh and Sigmoid activation functions.

In order to create a bigger convolutional neural network you have to reset the graph instance in a continuous environment.

```
import tensorflow as tf tf.
reset_default_graph()
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1],
name=‘input’)
convnet = conv_2d(convnet, 32, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 128, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 32, 5, activation=‘relu’)
convnet = max_pool_2d(convnet, 5)
convnet = fully_connected(convnet, 1024, activation=‘relu’)
convnet = dropout(convnet, 0.8)
convnet = fully_connected(convnet, 2, activation=‘softmax’)
convnet = regression(convnet, optimizer=‘adam’,
learning_rate=LR, loss=‘categorical_crossentropy’,
name=‘targets’)
model = tflearn.DNN(convnet, tensorboard_dir=‘log’)
model.load(MODEL_NAME)
print(‘model loaded!’)
train = train_data[:-500]
test = train_data[-500:]
X = np.array([i[0]
reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train] test
x = np.array([i[0] for i in test])
reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1]
model.fit({‘input’: X}, {‘targets’: Y}, n_epoch=3, validation_set=
({‘input’: test_x}, {‘targets’: test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

On the other hand, some of the disadvantages of the ReLU activation function are that output is not centered at zero and it is always positive. In addition, the gradient will be killed value x is less than zero. However, some techniques such as leaky ReLU as well as parametric ReLU are commonly used in order to overcome this as well as to encourage your model. Another disadvantage of ReLU activation function is that a gradient is not defined as value x that is equal to zero. However, this issue may be easily catered if you use sub-gradients and if you post some less practical challenges since challenges like value x is equal to zero are generally very rare.

In order to conclude on the topic of activation functions, ReLu is most commonly used in convolutional neural networks, and it is a great choice for you as well even if you a beginner. ReLU activation functions can be used efficiently when it comes to the convolutional neural networks, and here we will use it as well.

Data processing when it comes to the neural networks commonly uses steps mean centering and same size images. The same size images data processing step is when every image is converted to be matching in size to other images generally placed in squares. Mean centering on the hand is a step that involves a certain pixel. This step involves every pixel, and every pixel value and every image will be in fact obtained from a certain pixel. Sometimes it may happen, but really rarely that mean centering is done along the green, blue and red channels. It should be noted that normalization generally is not done with images.

When it comes to the initializing weights, there are various techniques that can be used and some of the most common techniques include all zeros, Gaussian random variable, and Xavier initialization. All zero weight initializing technique when it comes to the computer vision is generally a bad idea since in the case when you use it all neurons will generate the same output at the final stage as well as very similar gradients will flow back during the back-propagation.

The network in generally will not train adequately, so the outcomes are mostly undesirable. When it comes to the Gaussian random variable weight initializing technique, the weights are generally initialized with some random Gaussian distribution of zero mean and another standard deviation that is 0.1 to 1e minus five.

Gaussian random variable weight initializing technique works perfectly when it comes to the shallow networks, for instance, if you have five hidden layers you can use to. However, the Gaussian random variable technique is not used for deep networks. If you have a deep network, then small weight, in fact, make the outputs small as well as you are moving towards the end. So as you are moving the variables also become smaller. The gradient is killed at the end since the gradient becomes smaller as well as you move towards the end. It should be noted that you need to use standard deviation when it comes to the Gaussian distribution that will work well for your particular network.

Xavier initialization is another weight initializing technique that suggests some variance in the Gaussian various weights distribution used for every neuron. This variance state that every neuron, in fact, should depend on the certain number of different outputs in that particular layer. The recommended variance, in fact, is the square root of total inputs. Therefore, the number code in order to initialize certain weights will be the weights of particular layers with some n value on layer inputs.

There is one more thing that should be kept in mind while using Relu activation functions that are generally most commonly used. So remember that certain weights initialization may be in that form that some neurons may not get activated on time since there might be some negative input. This is something that you need to check before you go deeper with your project. It also may be surprising to you that around ten to twelve percent of the ReLU activation functions may be dead at some time while data training is in progress even if you are at the very end.

Computer Vision with Python

We already know that human vision sense in incredibly advanced. As soon as you open your eyes, you see every detail and object around you within a fraction of milliseconds, and we can detect and identify these objects instantly that are in our field of view unconsciously without any hesitation or previous thought.

We are also able to name every object that is around us as well as to perceive the depth of these objects with great accuracy, and we can perfectly distinguish objects' contours as well as to detect and identify various objects that are in the background. Our eyes take in some raw voxels of various color data, and our brain transforms this data into some relevant and meaningful information. So color information will be perceived as different lines, shapes curves and other which may indicate at what we are looking at like various objects, people, building, etc.

Programming machines for computer vision are created in order to with great accuracy replicate human vision sight that can be incorporated in various fields and therefore many major companies are used deep learning techniques and convolutional neural networks for great performance in facial recognition and object detection that is part of the computer vision field. Some major social network platform also uses this technique for automatic facial recognition like Facebook.

These techniques are also incorporated into self-driving cars that are the future. However, we already know what great computer vision challenges are, so tasks that are involved here might be very difficult to solve. Human automatically see objects, contours, line, and shapes but computers on the other hand just see a great collection of matrices and numbers.

In order to solve this problem and to learn more complex image features that are out of raw pixels and their values, convolutional neural networks are used. Moreover, a great place to start working on computer image is programming language Python providing you everything you will need on this journey.

Python is one of the most used programming languages for various purposes, and it will be the perfect destination for convolutional neural networks and tasks for computer vision. Convolutional neural networks are behind some great success when it comes to the computer vision as well as in the field of image and speech recognition. Therefore, we already know that convolutional neural networks use two fundamental structures feature maps and filters that are also called feature detectors.

These fundamental structures can be presented in the form of certain groups containing various neurons that will enable us to build different networks. So, for instance, we can say that we have one image and our main goal is to detect both vertical and horizontal edges. In order to achieve this, we have to create an individual convolutional filter.

A convolutional filter is, in fact, a very small matrix, which represents a certain feature that is relevant to us and that we want to find within the original image. The convolutional filter that is placed on top, in fact, tries to identify and detect various parts of that original image with some vertical lines. On the other hand, that convolutional filter that is placed at the bottom tries to detect and identify various parts of the image that contains some horizontal lines.

The actual process of detection will work by taking that convolutional of the certain filter with the image that is original. When we perform the convolution in the Python outputs of that convolution will locate various positions of different features within the original image, which will represent our features maps. In order to turn feature map filters into some concrete model in a convolutional neural network we will use hierarchy scheme.

In this certain scheme, a feed-forward neural network and all its layers containing neurons will represent both the feature map and the original image while the convolutional filters represent a specific combination of different neural network relationships. These relationships are replicated throughout the complete input.

So in your scheme that you have built a connection that is in the same color are in fact restricted to have the same weight always. In order to achieve this you have to initialize every relationship in that certain group with some identical weights and always you should averaging the certain weight updates of that collection before applying them at the end of every iteration.

Computer vision using an open-source Python library.

```
from pylab import
from numpy import
import pickle
from PCV.classifiers import bayes
from PCV.tools import imtools
open(‘../data/points_normal.pkl’, ‘r’)
class_1 = pickle.load(f)
class_2 = pickle.load(f)
labels = pickle.load(f)
bc = bayes.BayesClassifier()
bc.train([class_1,class_2], [1,-1])
open(‘../data/points_normal_test.pkl’, ‘r’)
class_1 = pickle.load(f)
class_2 = pickle.load(f)
labels = pickle.load(f)
print bc.classify(class_1[:10])[0]
points = vstack((x,y))
return bc.classify(points.T)[0]
```

```
imtools.plot_2D_boundary([-6,6,-6,6], [class_1,class_2], classify,
[1,-1])
show()
```

The output layer will be your feature map that is generated by that certain filter. A neuron in your feature map will be activated in the case when the filter is contributing to overall activity that detected some adequate feature at some corresponding position located in the previous layer.

In order to do more meaningful tasks, you can also create a feature map at some layer that will detect a various object or faces. For instance, if you have faced with nose, mouth and two eyes you will, in fact, need the information from all of these three different feature maps that will be represented at the convolutional neural network layers.

In other words, you are able to create filters which depend on some volumes of various information and they still might traverse in various feature maps within a single layer. These kinds of relationship also can be captures if you are using a full-fledged neural network, which would do just that.

When you are moving deeper into your convolutional network, you may need to sharpen information contained within your feature map. This is mainly because when certain feature is presented in some previous layer this may result in the feature map which will tend to contain some central hotspot that will be surrounded by a certain halo, so the feature map often may be weakly or not completely detected so you might need to sharpen information included in the layer.

In order to achieve this, you will use a technique that is called a max pooling. Using this technique, you will divide your feature map into certain disjoint squares or titles, and further you will take an action that is a maximum of all neurons that are situated at each tile. Max pooling for the feature maps is great when it comes to the detection process that will be clearer by removing some not relevant and uninformative halos.

Max pooling also reduces the total number of various parameters in your convolutional neural network. Max pooling is also combating some potential issues in over-fitting.

If you put all these concepts together, you are ready to start working on some more complex computer vision problem. For instance, let’s imagine that we have some blood smear from a patient and our main goal is to detect what is the potential for malaria invasion as well as to diagnose what stage of infection is present in that patient. The stage of infection may be some early stage or some late stage.

The possibility is also that there is no any infection present in a patient. You can create convolutional neural networks that will contain four layers, which would be followed, by classic dense feed-forward networks that end in a three-way soft max that is providing a great confidence in all of these three possible outcomes, early stage, late stage and no infection at all.

Just like traditional convolutional feed-forward network, your convolutional neural network will be trained by using stochastic gradient descent as well as you will use a dropout in your dense layer in order to prevent over-fitting.

Therefore, you will further tune you convolutional neural network for Malaria infection stage, and preliminary results will indicate that this neural network performs much better than the classic approaches to machine learning like Bayesian classification techniques or vector machines.

In addition, some of the most recent works done by Baidu and Google really indicate that convolutional neural networks are absolutely logins great accuracies and at some tasks, even accuracies that are better than humans are. It really seems that convolutional neural networks are able to be of great influence for some of the futuristic technology that is just around the corner.

An example of max pooling.

matrix:

array([[20, 200, -5, 23],

[-13, 134, 119, 100],

[120, 32, 49, 25],

[-120, 12, 09, 23]]) kernel: 2 x 2

soln:

array([[200, 119], [120, 49]])

Chapter 3 Image Recognition

TensorFlow Image Recognition

A TensorFlow is one of the most popular Python libraries and a perfect place for building various neural networks since it provides a high-level API, which makes it very easy to construct any neural network including convolutional neural networks as well. TensorFlow provides great methods, which facilitate the design of fully connected or dense convolutional layers at the same time adding various activation function as well applying different dropout regularization.

Image recognition using TensorFlow.

```
cd models/tutorials/image/

imagenet python classify_image.py

giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca
(score = 0.88493)

 indri, indris, Indri indri, Indri brevicaudatus (score = 0.00878)

lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens
(score = 0.00317)

custard apple (score = 0.00149)

earthstar (score = 0.00127)

bazel build tensorflow/examples/label_image

bazel-bin/tensorflow/examples/label_image/label_image

I tensorflow/examples/label_image/main.cc:206] military uniform
(653): 0.834306

I tensorflow/examples/label_image/main.cc:206] mortarboard
(668): 0.0218692
```

```
I tensorflow/examples/label_image/main.cc:206] academic gown
(401): 0.0103579

I tensorflow/examples/label_image/main.cc:206] pickelhaube
(716): 0.00800814

I tensorflow/examples/label_image/main.cc:206] bulletproof vest
(466): 0.00535088

bazel-bin/tensorflow/examples/label_image/label_image —
image=my_image.png

string input_name = “file_reader”;

string output_name = “normalized”;

tensorflow::Node* file_reader =
tensorflow::ops::ReadFile(tensorflow::ops::Const(file_name,
b.opts())

b.opts().WithName(input_name))

tensorflow::GraphDef graph

TF_RETURN_IF_ERROR(b.ToGraphDef(&graph))
```

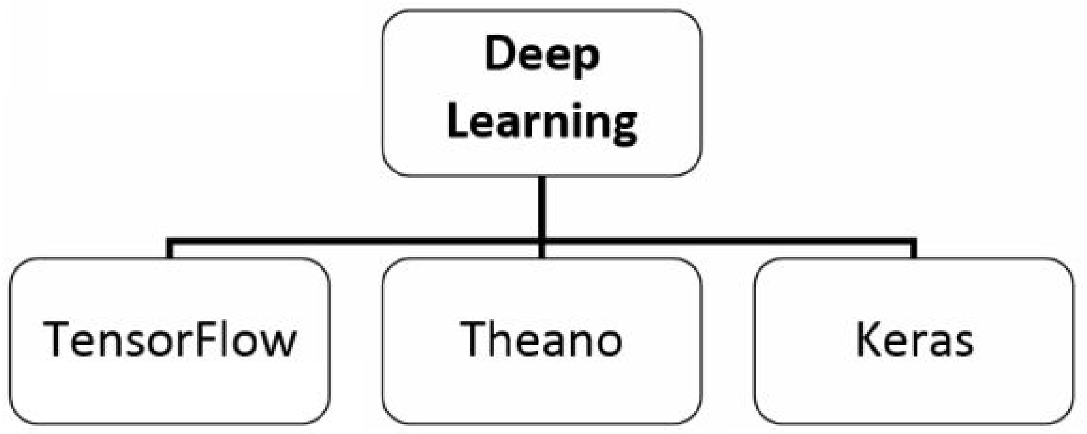
Image Recognition:

- *Image*
- *Sub-images of individual parts of the image*
- *Binary representations of the sub-images, zero is represented as white, and one is represented as black*
- *Supervised neural networks which has been trained in order to recognize images*
- *Neural network output and all recognized part of the image*

In order to start and create a convolutional neural network you will have to create a file that will be called as cnn-mnist.py and some further codes have to be incorporated as well. You will import some absolute import feature as well as import division and print functions. You will also import numpy as np and import TensorFlow as well. Further codes include form TensorFlow contribution to import learn and TensorFlow contribution to Python learn. As soon as you finished with importing these codes, you will see your application logic.

In order to use convolutional neural networks for image recognition you have to download already trained model form TensorFlow library. In order to download it, you will need around 200 megabyte of free space on your hard disk. The command classify image will run for the first time and the next step is to clone TensorFlow algorithm from GitHub. As soon as you run commands from GitHub, you will see the picture that you provided.

If your model is running accurately, you will see a certain script that will show that particular output of pictures that you provided. You are also able to edit the imported pictures using command image file. In case when you download the model data to some other directory, you have to print command model dir to the previous directory that you used.



Convolutional Layers in Theano

The convolution operator in Theano, another very popular Python library is, in fact, the main workhouse used in order to implement various convolutional layers. The convolutional operator or simply ConvOp will be used in Theano when you run arguments theano tenwor signal convolutional two-dimensional layer. Two inputs will appear. One of them is a four-dimensional tensor that is corresponding to a mini-batch of that input image.

The convolutional operator in Theano.

```
import theano
from theano import tensor as T
from theano.tensor.nnet import conv2d
import numpy
rng = numpy.random.RandomState(23455)
input = T.tensor4(name=‘input’)
w_shp = (2, 3, 9, 9)
w_bound = numpy.sqrt(3 * 9 * 9)
W = theano.shared( numpy.asarray( rng.uniform( low=-1.0 /
w_bound, high=1.0 / w_bound, size=w_shp)
dtype=input.dtype), name =‘W’)
b_shp = (2,) b = theano.shared(numpy.asarray(
rng.uniform(low=-.5, high=.5, size=b_shp), dtype=input.dtype)
name =‘b’)
conv_out = conv2d(input, W)
output = T.nnet.sigmoid(conv_out + b.dimshuffle(‘x’, 0, ‘x’, ‘x’))
f = theano.function([input], output)
```

Another input is a four-dimensional tensor that is corresponding to the various weight matrices denotes commonly as W. In order build convolutional neural networks using Theano you will follow commands import theano and from Theano to import tensor. The final input will be incorporated into the convolutional two-dimensional layer. The next step is to import numpy. Numpy will be equal to rng or some random state.

Further commands include initializing four-dimensional tensor in order to get input. Then you will initialize some shared variables as well as variables for weights. The next step is to initialize some shared variable for vector bias that is a one-dimensional tensor. It should be noted that biases are commonly initialized to a value of zero. Nevertheless, in this very same application, we can simply apply our convolutional layer to any image without previously learning image parameters. Therefore, in fact, we initialize them to some random values that will stimulate learning.

The further steps include importing Theano shared numpy arrays. Your next step is to build some symbolic expression that will compute the convolutional layers of input with some filters. So the argument for this will be convolution out is equal to convolutional two-dimensional layer. The next step is to build some symbolic expression, which would be added to bias, so it will apply certain activation function as well as it will create some new neural network output.

In addition, you should use one powerful tool called dimshuffle that will do all the reshaping in any tensor. Dimshuffle tool when used will allow you to shuffle various dimension around as well as to insert some new dimensions that will go along with a tensor so that the tensor will be completely broadcastable. This tool will work perfectly on three-dimensional convolutional layers that are not broadcastable regarding dimensions.

Therefore, the first tensor dimension will be completely broadcastable. Therefore, the first dimension will be completely broadcastable, so you will also have the third dimension of that input tensor just like the second dimension of that resulting tensor, etc. If your tensor has a shape like twenty, thirty, forty, in that case, your resulting tensor will have certain dimensions of shape one, forty, one, twenty, thirty.

Max-Polling in Theano

Another very important concept of convolutional neural networks when it comes to the image recognition is the concept of max-pooling. Max-pooling, in fact, is a type of nonlinear sampling where some partitions of the input image are set in certain non-overlapping squares and for every square region, there are some maximum value outputs.

Maximum pooling concept comes to be very useful in vision for various reasons including it eliminates values that are not maximal as well as it reduces various computation for higher levels. Another reason why max-pooling is of great importance is that it provides a certain form of translation invariance. For instance, imagine some cascading max-pooling layer together with some convolutional layer. In this case, there are eight possible directions in that one is able to translate the input image just by a single pixel.

If you do max-pooling over two by two regions, three of these eight possible configurations will provide the same output at your convolutional layer. In a case when you perform max polling for three by three square region, this will jump to five of possible eight configurations.

Max-pooling, in fact, is a smart way to reduce the dimensions of some intermediate representation since max-pooling provides some additional robustness to every position. In order to initiate the concept of max-pooling in theano, you have to initiate a theano tensor signal two-dimensional pool argument. This function will be taken as a certain input of some N dimensional tensor and N is greater than two. There is also a downscaling factor as well as max-pooling in the case where there is two trailing dimension of some tensor.

theano.tensor.signal.pool.pool_2d

theano.tensor.signal.pool.max_pool_2d_same_size

theano.tensor.signal.pool.pool_3d

Further steps toward initializing max-pooling include command input tensor, and the max-pooling shape will be equal to two. Pool out functions will be equal to pool two dimensional so you have to input max-pooling shape as well as a command to ignore border shape. Invals will be equal to random numpy in a random state. The next step is to set print with ignoring border that will be set as true. The next steps are print invites and print output.

Further, you will initiate commands pool out that will be equal to two dimensional pool and this time ignore border will be set to false. As soon as you are done, you will see generated output. It should be noted that this company max pool two dimensions in Theano is slightly different from other commands since it requires the downscaling factors. In this case, the tuple will be in length two at the same time containing downscaling factors together with image height and width. These factors will be known when it is time to build a graph.

MNIST Vision Dataset

MNIST is very simple computer dataset that contains thousands of handwritten digits in numerical form. MNIST data set also includes some labels for every image that tells us which digit is used for. Therefore, in this part of the book, we will see how to train a certain model that will look at a certain image and predict digits that are contained in that image.

Our goal in this task is not to elaborate model that will achieve a great performance since that is a task for an expert in this field. We will dip into a tensor flow and MNIST dataset initiated from here starting with a very simple model known as softmax regression.

Loading the MNIST dataset.

```
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from urllib
import urlretrieve
import cPickle as pickle
import os
import gzip
import numpy as np
import theano
import lasagna
from lasagna
import layers from lasagne.updates
import nesterov_momentum
from nolearn.lasagne
import NeuralNet
from nolearn.lasagne
import visualize
from sklearn.metrics
import classification_report
from sklearn.metrics
import confusion_matrix
```

Define your MNIST loading function.

```
def load_dataset():
filename = ‘mnist.pkl.gz’
if not os.path.exists(filename):
print(“Downloading MNIST dataset…”)
data = pickle.load(f)
```

```
X_train, y_train = data[0]
X_val, y_val = data[1]
X_test, y_test = data[2]
X_train = X_train.reshape((-1, 1, 28, 28))
X_val = X_val.reshape((-1, 1, 28, 28))
X_test = X_test.reshape((-1, 1, 28, 28))
y_train = y_train.astype(np.uint8)
y_val = y_val.astype(np.uint8)
y_test = y_test.astype(np.uint8)
return X_train, y_train, X_val, y_val, X_test, y_test
```

Inspect the MNIST dataset.

```
X_train, y_train, X_val, y_val, X_test, y_test = load_dataset()
plt.imshow(X_train[0][0], cmap=cm.binary)
```

The actual code to achieve what is our goal here is very simple, and some great stuff can happen in just three lines how much is needed to perform this task. Here it is important to get a better insight into how TensorFlow works and what are fundamental deep learning concepts.

MNIST Dataset:

Not all arrays are MNIST digits: randomly pick a few points, each pixel is randomly black, white or some shade of gray, you most likely will get a noisy image

The dataset is split into three mutually exclusive sub-sets: training data contains 55000 images that are used to train algorithm, test data contains 10000 images that are used to test the algorithm and validation that contains 5000 images that are used to optimize the algorithm

In machine learning, you need separated data: to make sure that what you have learned actually generalizes

Test data: used to test the algorithms, not to improve or optimize the algorithm

If you are copying and pasting MNIST codes from some website, then look for the initial code from tensor flow to mnist import data. MNIST will be equal to input data, and one hot will be true. The MNIST computer visual dataset consists of three parts, and fifty-five thousand data points are training data that you initialize by command mnist train.

Ten thousands point of mnist data are test data, and you initialize it by command mnist. Finally, five thousand points are validation data that you initialize by command mnist validation. This split of MNIST data is very imports since in deep learning it is of great significance to have some separate data that we, in fact, do not learn from therefore we can be sure that things that we have learned in fact generalize.

Every MNIST data point, in fact, has two parts, a first part is a certain image containing a handwritten digit, and other is a corresponding label. It may be easier to understand if we call images as x and labels as y. Both the test set and training set contain particular images as well as corresponding labels consisted within.

For instance, training images are argument mnist to train images, and training labels are mnist to train labels. If we imagine that we have an image that contains twenty-eight pixels by twenty-eight pixels, we are able to interpret this as a large array consisted of numbers.

The first step will be flattening of this array into a certain vector that will contain seven hundred eighty-four numbers since we multiplied twenty-eight by twenty-eight. It is not relevant how you flatten this specific area, as long as you are consistent between various images. If we look at images from this certain perspective, then the MNIST images are just a wide collection, so various points contained in seven hundred eighty-four dimensional vectors spaces containing very rich structure.

When you flatten that certain data you will, in fact, throw away information about the two-dimensional structure of your image. This is necessarily a bad thing since computer vision techniques exploit data structure. However, simple softmax regression will not do this to a data structure.

The further result will be that mnist train images are a tensor or dimensional array in shape of 55000, 784. The first dimension of tensors is, in fact, an index into the collection of images. The second dimension will be the index for every pixel that is contained in that image. Each entry you made in the tensor, in fact, is a pixel of intensity between zero and one for a certain pixel in a certain image.

Every image in MNIST dataset has some corresponding label that is a number between zero and nine, which are representing the digits. On the other hand, one hot vector is a vector that is zero in almost every dimension and one in one dimension. As soon as you get to this part, you are ready to make your model.

We already know that MNIST dataset is a collection of handwritten digits that are between zero and nine. Therefore, there are only ten possibilities in terms what that image may be. Our goal is to look at a certain image and give all probabilities for that certain image.

For instance, your picture may look like a picture of nine, so you are eighty percent sure that it is a nine, but there is also a slight chance that the picture might be eight since there is the top loop. There is also a small probability for other options since you are not hundred percent sure. This is a very common case in which softmax regression is a logical model to be used.

Softmax is really a thing to do when you want to assign various probabilities to some object when there are several different things. Softmax gives us a collection of all values between zero and one, which adds up to one. Further, you are able to train some more complicated models as well. In this case, the final layer will be a layer of softmax.

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets(“MNIST_data/”, one_hot=True)
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), 
reduction_indices=[1]))
train_step = 
tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
for _ in range(1000):
batch_xs, batch_ys = mnist.train.next_batch(100)
sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: 
mnist.test.labels}))
```

Softmax regression involves two steps, you first have to add up some evidence of your input in particular cases, and the next step is to convert that particular evidence into various probabilities. The further step is to perform a weighted sum of all pixel intensities. In the case where the weight is negative, that means that pixel having a higher intensity is, in fact, evidence against that certain image that is contained in a particular class.

On the other hand, weight is positive if it is evidence that is in favor. Further steps will be adding some extra evidence that is called as bias. We commonly add this additional evidence in order to say that some things might be more likely separate of the input. So think of a softmax regression as exponentiation of the model inputs and further normalizing them.

If you look at softmax regression from this perspective, the exponentiation really means that multiple units of evidence will increase the weight that is given to any hypothesis. On the other hand having one loss element of evidence will mean that your hypothesis gets a certain fraction of it is at an earlier weight. It should be noted that there is no hypothesis that has negative or zero weight. Softmax, in fact, is used to normalize these weights so when they add up to each other, they form an accurate probability distribution.

Chapter 4 Convolutional Filters

Convolutional neural networks are proven to be of great significance when it comes to the complex image object detection and recognition, and there was the word about that in the previous chapters. This chapter of the book will introduce you to some of the most common convolutional filters that are used widely in image recognition tasks.

Commonly used Convolutional Layers:

- *Convolution layer: This layer convolves some input image with a collection of filters that are easily learned, and each of the filters will produce a single feature map at the final stage*
- *Pooling: Max-pooling sub-elements of the input image transfer into a collection of some non-overlapping squares and for every square area or also called sub-region outputs are the maximum*
- *Rectified-linear or ReLU: If you have an output of some value x, this ReLU layer will compute the output as value x is greater than zero and there is also a negative slope when value x is less than zero*
- *Fully connected layer or inner productivity: In this layer, every image is created as vector and every vector point contributes to every single point of the vector that is product of output*

As soon as your import these four layers to the Python library, a lot of forwarding networks is available in order to be implemented in Python framework. You can use commonly used and very respected convolutional neural network ImageNet that contains already trained and implemented benchmarks and datasets.

As soon as you start ImageNet, you will see what are different convolutional layers that are required in ImageNet. If you are using something rather than ImageNet, the results will be the same, and the overall process is same as well.

For instance, if you have five convolutional layers and three fully connected layers that these layers will occupy ninety-nine percent of the overall time that is needed for this certain network. You will find that there are three different convolutional filter sizes that are available for different convolution layers.

ImageNet Classification with Python.

```
model = VGG16(weights="imagenet")
preds = model.predict(preprocess_input(image))
print(decode_predictions(preds))
pip install pillow
import keras
keras.__version__ '1.0.6'
git clone https://github.com/fchollet/deep-learning-models
from keras.preprocessing import image as image_utils
from imagenet_utils import decode_predictions
from imagenet_utils import preprocess_input
from vgg16 import VGG16
import numpy as np
import argparse
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-i", "—image", required=True, help="path to the input image")
args = vars(ap.parse_args())
orig = cv2.imread(args["image"])
print("[INFO] loading and preprocessing image…")
```

```
image = image_utils.load_img(args[“image”], target_size=(224,
224))
image = image_utils.img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
print(“[INFO] loading network…”)
model = VGG16(weights=“imagenet”)
print(“[INFO] classifying image…”)
preds = model.predict(image) (inID, label) =
decode_predictions(preds)[0]
print(“ImageNet ID: {}, Label: {}”.format(inID, label))
cv2.putText(orig, “Label: {}”.format(label), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
cv2.imshow(“Classification”, orig)
cv2.waitKey(0)
P = decode_predictions(preds) (imagenetID, label, prob) = P[0][0]
```

The computational time, in fact, varies from one convolution layer to another. The computational time also depends on different filters that are applied as well as on the size of your image. Since the computational time vary, that creates various layers that are in fact optimized to different layers, so it’s inefficient.

In order to avoid this filter three by three or the smallest filter will be used as a basis for more complex convolutional layers. These smallest layers are in fact the most demanding when it comes to their computing due to their number of input as well as their number of output features that will eventually be processed with them.

You can represent more complex filter sizes like multiple passes of that smaller three by three filters. However this will add some more inefficiency in the overall processing, but at the same time, it will allow you to use logic reuse among various layers.

When it comes to the convolutional filters that are commonly used, one of that kind is a response to an entire signal. It means that the response of the overall system will be with an inpulse response. In other words, we can say that response to an entire single is simply the convolution of input denoted as x and impulse response denoted as h.

When it comes to some properties of convolutions, it should be noted that convolution properties are greatly similar to mathematical properties, so those are commutative, associative, distributive over addition and derivative.

The convolutional Gaussian filter is when the maximum value is equal to one. On the other hand, normalized Gaussian is when a total area is equal to one. You are also able to convolve a Gaussian with another Gaussian.

When it comes to using convolutional neural networks in noise recognition, first we have to distinguish what is noise. So noise, in fact, is anything that is not some signal. Singal, in fact, carries various information that may be relevant and useful to us. Therefore, noise is anything that's not a signal, so noise does not carry any relevant information. Noise, on the other hand, may be completely random either spatially and temporally.

Noise is also structured, and it structures randomness. When it comes using convolutional neural networks and convolution filters, for instance, adding echo effects to some noise, you will have mean that is the expected or average value. You will also have a variance that is the expected value of some squared error. The standard deviation is the square root of a variance.

On the other hand, when it comes to the using of convolutional neural networks for adding image filters you have to consider any image as a collection of random variables that we use in order to ensemble an image from the space that contains all possibilities. This collection or ensemble of images will have an average or mean. Also if you sample a sufficient number of images, the ensemble average approaches that original noise-free signal.

Another convolution filter is a signal to noise ratio that is widely used in sound detection and recognition. Therefore, if you compare the strength of your image or signal to the certain variance between all individually acquired images, you will get this signal to noise ratio. The better or higher the signal to noise ratio the better is your ability regarding discerning various signal information.

Noise and frequency domain, on the other hand, is slightly different. When you use it, keep in mind that white noise always has equally random values of all possible frequencies. On the other hand, colored noise has an amount that is not equal to various frequencies. It should be noted that signal mostly has lower frequencies rather than high, so the effect of certain white noise commonly is greatest for higher frequencies.

Convolutional Filters:

- *Low pass filter: this filter eliminates all high frequencies and leave only low frequencies*
- *High pass filter: this filter eliminates all low frequencies and leave only high frequencies*
- *Band pass filter: limits frequencies that will remain*
- *Gaussian smoothing: has the effect of cutting off only high-frequency elements consisted in frequency spectrum*

Convolution low-pass filter recalls all quick changes that happen in a signal or in an image that requires some high frequencies. Commonly high-frequency details are buried in some noise that also requires only high frequencies.

There are some methods that are used to reduce noise and most common is pixel average. This method, in fact, same average pixel over some multiple images that are contained in the same scene. It also averages multiple pixels that are neighboring contained in a single image.

On the other hand convolution filter average use square function, Gaussian, and box filter to locally average the image and signal. Therefore, square or box function is uniformly averaging, and Gaussian is weighted in center averaging. Both of these can be used in order to blur image or signal.

Convolution filter low pass filter is also known as spatial blurring. Any convolution layer that contains all positive weights, as well as all negative weights, does spatial blurring, low-pass filtering, and weighted averaging so we can say that all of them are entirely equivalent.

There are two common ways to think of convolutional filters, and those are spatial and frequency. Spatial filtering is, in fact, a convolution by some kernel that is spatial-domain. On the other hand frequency is multiplication by a filter that is frequency-domain. Both of these can analyze and implement.

When it comes to the low-pass filter, it should be noted that it reduces noise, but at the same time, it blurs an image. It means the worse noise you got, the more blur you need to remove it. Blurring is low-pass convolutional filtering, so de-blurring is opposite of that high-pass filtering. Blurring so explicit high-pass filtering as well as unsharp masking. Blurring is also used in common edge detection. The tradeoff blurring filter is that it reduces blur, but at the same time, it increases noise.

Other commonly used convolution filtering technique is unsharp masking used for high-boost filtering. So you use unsharp masking when you want to sharpen an image or signal and to subtract some of the blurred input. Unsharp masking procedure includes blurring the image, subtracting from the original, multiplying some weighting factors and adding it back to the original image.

Conclusion

You came to the end of this book, and I am confident that you have learned valuable information that will significantly improve your understating of deep learning and deep neural networks in general. This book is focused on convolutional neural networks, so you learned what great opportunities these neural networks have to offer you.

This book, in general, can be of great for you since you are interested in a field of deep learning. Deep learning in fact greatly affected various scientific fields and offered new opportunities and greater techniques that are proven to be of great significance in various fields including natural language processing, speech and image recognition, computer vision, business management and other numerous fields. When it comes to the convolutional neural network, which is a great focus of this book, you have learned how to implement the convolutional neural network in computer vision, image, and sound recognition.

We are aware of the fact what great impact in various fields the introduction of convolutional neural networks has brought. Many major companies including Microsoft, IBM, NVIDIA have been using deep learning techniques for various tasks, and deep learning replaced traditional machine learning techniques.

Deep learning brought various technologies to a completely new level with various neural network techniques and algorithms. Deep learning is among the most active areas when it comes to the research, and most certainly, it is paving the way for some greater learning techniques.