Application of Machine Learning and Neural Networks to Sentiment Analysis

Summer Internship Project

Report 1

Sreyashi Bandyopadhyay

<sreyashiban14@gmail.com>

Supervised By: Soumotanu Mazumdar

Date: 8th May 2021

STUDENT PROFILE

Name: Sreyashi Bandyopadhyay

College: Heritage Institute of Technology

Department –Applied Electronics and Instrumentation

Year of Study- 2nd year, 4th semester

Year of Passing- 2023

Internship Application Id- WB/HIT/965

Abstract

Sentiment analysis (or opinion mining) is a natural language processing technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on movie reviews to help users decide if the movie is worth their time. A summary of all reviews for a movie can help users make this decision by not wasting their time reading all reviews. Movie-rating websites are often used by critics to post comments and rate movies which help viewers decide if the movie is worth watching. Sentiment analysis can determine the attitude of critics depending on their reviews.

Sentiment analysis of a movie review can rate how positive or negative a movie review is and hence the overall rating for a movie. Therefore, the process of understanding if a review is positive or negative can be automated as the machine learns through training and testing the data.

This project aims to rate reviews using traditional machine learning classifiers (Naïve Bayes) along with neural networks based on Deep Learning and compare which gives better and more accurate results.

Classification is a data mining methodology that assigns classes to a collection of data in order to help in more accurate predictions and analysis. The classification would be binary , that is the movie reviews would be classified as either positive or negative. The IMDB 50K movie review dataset has been used for this purpose.

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I would like to express my heartfelt thanks and gratitude to my supervisor Mr. Soumotanu Mazumdar of National Institute for Industrial Training who gave me the golden opportunity to do this project and guided me in an exemplary manner. It helped me in doing a lot of research and I came to know about a lot of things related to this topic.

Last but not the least I thank my friends and my family for their wholehearted support.

1. Introduction

1.1 Motivation

Movie reviews are an important way to gauge the performance of a movie. While providing a numerical/ stars rating to a movie tells us about the success orfailure of a movie quantitatively, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie. A textual movie review tells us about the the strong and weak points of the movie and deeper analysis of a movie review can tell us if the movie in general meets the expectations of the reviewer. Sentiment Analysis is a major subject in machine learning which aims to extract subjective information from the textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. It can be used to determine the attitude of the reviewer with respect to various topics or the overall polarity of review.

In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. We aim to utilize the relationships of the words in the review to predict the overall polarity of the review.

1.2 Dataset

The dataset used for this task was collected from Large Movie Review Dataset which was used by the AI department of Stanford University for the associated publication. The dataset contains 50000 training examples collected from IMDB where each review is labeled with the rating of the movie on scale of 1-10. As sentiments are usually bipolar like good/bad or happy/sad or like/dislike, we categorized these ratings as either1 (like) or0 (dislike) based on the ratings.

After this a sample size of 10000 movie reviews were taken for the training purpose. Classical Machine Learning Models like Naïve Bayes and neural networks, namely ANN , CNN and RNN were applied to check the model performance and create a comparison between them .

Attached below is an illustration of how the movie reviews look in the dataset.

Positive	Negative
GREAT movie and the family will love it!! If kids are bored one day just pop the tape in and you'll be so glad you did!!! ~~~~Rube i luv raven-s!	The script for this movie was probably found in a hair-ball recently coughed up by a really old dog. Mostly an amateur film with lame FX. For you Zeta-Jones fanatics: she has the credibility of one Mr. Binks.
Did Sandra (yes, she must have) know we would still be here for her some nine years later? See it if you haven't, again if you have; see her live while you can.	I would love to have that two hours of my life back. It seemed to be several clips from Steve's Animal Planet series that was spliced into a loosely constructed script. Don't Go, If you must see it, wait for the video ...
Verry classic plot but a verry fun horror movie for home movie party Really gore in the second part This movie proves that you can make something fun with a small budget. I hope that the director will make another one	This is without a doubt the worst movie I have ever seen. It is not funny. It is not interesting and should not have been made.

1.3 Problems with the data

The main problem encountered while dealing with user movie reviews were the fact that it was not processed properly to be able to be fed into ML based algorithms. Feeding them directly to the algorithm would be detrimental to model performance.

Hence the preprocessing step was necessary to get the data ready to work with.

1.4 Data Preprocessing

The data preprocessing for this project can be divided into the following subparts.

1. Removing HTML tags.
2. Removing special characters not of importance to sentiment analysis ,e.g. (-*%/?!) etc
3. Removing punctuation marks. (. ,)
4. Converting every character to lowercase for ease of evaluation

5. Removing words of less importance that do not contribute to a sentiment for e.g.- I , me ,my , the ,is was etc.

6. Stemming - Stemming is a technique used to extract the base form of the words by removing affixes from them.

7. Tokenization of text - The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

2. About the Project

This project has been performed to automate the task of classifying a movie review as positive or negative so that a user can decide whether or not to watch it.

Firstly , the data has been thoroughly pre-processed as described in section 1.4 . After that a new file has been made with the formatted data. Next that data has been fed into machine learning algorithms and neural networks to compare and contrast the model performance .

The algorithms used are as follows -

1. Gaussian Naïve Bayes
2. Bernoulli Naïve Bayes
3. Multinomial Naïve Bayes
4. Artificial Neural Network
5. 1 Dimensional Convolutional Neural Network
6. Recurrent Neural Network

2.1 Objectives

The sole objective of the project is to predict, using machine learning, the sentiment of a movie and train a model for that purpose.

Different techniques have been applied for this purpose and a comparative study has been made between the different accuracies shown by the various models.

The steps involved are -

1. DATA MINING- Data mining is a process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics and database systems Data mining is an interdisciplinary subfield of computer science and statistics with an overall goal to extract information (with intelligent methods) from a data set and transform the information into a comprehensible structure for further use.

2. DATA CLEANING-Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

3. DATA PREPROCESSING- Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

4. EXPLORATORY DATA ANALYSIS - In statistics, exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.

5. DIVIDING INTO TRAINING AND TESTING SET- Data splitting is the act of partitioning available data into two portions; usually for cross-validatory purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance.

6. APPLYING VARIOUS CLASSIFICATION MODELS- Various classification models were used to predict the probability of stroke in patients , after taking into consideration various other factors.

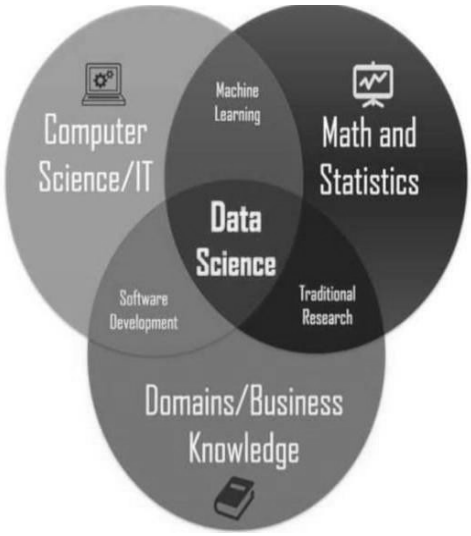
3. General Introduction to Relevant Topics

- **Data Science** - Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses complex machine learning algorithms to build predictive models.

The data used for analysis can be from multiple sources and present in various formats. Data science or data-driven science enables better decision making, predictive analysis, and pattern discovery.

Data science can add value to any business who can use their data well. From statistics and insights across workflows and hiring new candidates, to helping senior staff make better-informed decisions, data science is valuable to any company in any industry.

By extrapolating and sharing these insights, data scientists help organizations to solve vexing problems. Combining computer science, modeling, statistics, analytics, and math skills— along with sound business sense— data scientists uncover the answers to major questions that help organizations make objective decisions.



- Machine Learning - Machine learning is a type of technology that aims to learn from experience. For example, as a human, you can learn how to play chess simply by observing other people playing chess. In the same way, computers are programmed by providing them with data from which they learn and are then able to predict future elements or conditions.

There are various steps involved in machine learning:

1. collection of data
2. filtering of data
3. analysis of data
4. algorithm training
5. testing of the algorithm
6. using the algorithm for future predictions

Machine learning uses different kinds of algorithms to find patterns, and these algorithms are classified into two groups:

- supervised learning
- unsupervised learning

Supervised Learning

Supervised learning is the science of training a computer to recognize elements by giving it sample data. The computer then learns from it and is able to predict future datasets based on the learned data.

For example, you can train a computer to filter out spam messages based on past information.

Supervised learning has been used in many applications, e.g. Facebook, to search images based on a certain description. You can now search images on Facebook with words that describe the contents of the photo. Since the social networking site already has a database of captioned images, it is able to search and match the description to features from photos with some degree of accuracy.

There are only two steps involved in supervised learning:

- training

- testing

Some of the supervised learning algorithms include:

- decision trees
- support vector machines
- naive Bayes
- k-nearest neighbor
- linear regression

Python –

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object- oriented , imperative,functional programming, and procedural styles. It has a large and comprehensive standard library. Two major versions of Python are currently in active use:

Python 3.x is the current version and is under active development. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

3.1 Imports

The libraries that have been imported for this project are stated as follows.

1. Numpy- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2. Seaborn - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3. Pandas- In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

.

4. TensorFlow -TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming.

5. Keras- Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

6. Scikit learn- The Sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering .

7. NLTK - NLTK (Natural Language Toolkit) is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

3.2 Programming Language

Advantages of Python

1. Easy Syntax
2. Readability
3. High-Level Language
4. Object-oriented programming
5. It's Open source and Free
6. Cross-platform
7. Widely Supported
8. It's Safe
9. Extensible

Easy Syntax of Python

Python's syntax is easy to learn, so both non- programmers and programmers can start programming right away**.**

Very Clear Readability of Python

Python's syntax is very clear, so it is easy to understand program code. (Python is often referred to as "executable pseudo-code" because its syntax mostly follows the conventions used by programmers to outline their ideas without the formal verbosity of code in most programming languages.

Python High-Level Language

Python looks more like a readable, human language than like a low-level language. This gives you the ability to program at a faster rate than a low-level language will allow you.

Python Is Open-Source and Free

Python is both free and open-source. The Python Software Foundation distributes pre -made binaries that are freely available for use on all major operating systems called CPython. You can get CPython's source-code, too. Plus, we can modify the source code and distribute as allowed by CPython's license.

Python is a Cross-platform

Python runs on all major operating systems like Microsoft Windows, Linux, and Mac OS X.

Python Object-oriented programming

Object-oriented programming allows you to create data structures that can be reused, which reduces the amount of repetitive work that you'll need to do.

Programming languages usually define objects with namespaces, like class or def, and objects can edit themselves by using keyword, like this or self.

Most modern programming languages are object-oriented (such as Java, C++, and C#) or have support for OOP features (such as Perl version 5 and later). Additionally, object-oriented techniques can be used in the design of almost any non-trivial software and implemented in almost any programming or scripting language.

Python Widely Supported Programming Language

Python has an active support community with many websites, mailing lists, and USENET "net news" groups that attract a large number of knowledgeable and helpful contributors.

Python is a Safe

Python doesn't have pointers like other C-based languages, making it much more reliable. Along with that, errors never pass silently unless they're explicitly silenced. This allows you to see and read why the program crashed and where to correct your error**.**

Python Batteries Included Language

Python is famous for being the "batteries are included" language. There are over 300 standard library modules which contain modules and classes for a wide variety of programming tasks.

For example, the standard library contains modules for safely creating temporary files (named or anonymous), mapping files into memory (including use of shared and anonymous memory mappings), spawning and controlling sub-processes, compressing and decompressing files (compatible with gzip or PK-zip) and archives files (such as Unix/Linux "tar").

Accessing indexed "DBM" (database) files, interfacing to various graphical user interfaces (such as the TK toolkit and the popular WxWindows multi-platform windowing system), parsing and maintaining CSV (comma- separated values) and ".cfg" or ".ini" configuration files (similar in syntax to the venerable WIN.INI files from MS-DOS and MS-Windows), for sending e-mail, fetching and parsing web pages, etc. It's possible, for example, to create a custom web server in Python using less than a dozen lines of code, and one of the standard libraries, of course.

Python is Extensible

In addition to the standard libraries there are extensive collections of freely available add-on modules, libraries, frameworks, and tool-kits. These generally conform to similar standards and conventions. For example, almost all of the database adapters (to talk to almost any client-server RDBMS engine such as MySQL, Postgres, Oracle, etc) conform to the Python DBAPI and thus can mostly be accessed using the same code. So it's usually easy to modify a Python program to support any database engine.

.

3.2 Future Scopes of Python

Python is one of the fastest growing languages and has undergone a successful span of more than 25 years as far as its adoption is concerned. This success also reveals a promising future scope of python programming language.
In fact, it has been continuously serving as the best programming language for application development, web development, game development, system administration, scientific and numeric computing, GIS and Mapping etc.

Popularity of python

The reason behind the immense popularity of python programming language across the globe is the features it provides. Have a look at the features of python language.

(1) Python Supports Multiple Programming Paradigms Python is a multi-paradigm programming language including features such as object-oriented, imperative, procedural, functional, reflective etc.

(2) Python Has Large Set Of Library and Tools

Python has very extensive standard libraries and tools that enhance the overall functionality of python language and also helps python programmers to easily write codes. Some of the important python libraries and tools are listed below.

- Built-in functions, constants, types, and exceptions.
- File formats, file and directory access, multimedia services.
- GUI development tools such as Tkinter
- Custom Python Interpreters, Internet protocols and support, data compression and archiving, modules etc.
- Scrappy, wxPython, SciPy, matplotlib, Pygame, PyQT, PyGTK etc.

(3) Python Has a Vast Community Support

This is what makes python a favorable choice for development purposes. If you are having problems writing python a program, you can post directly to python community and will get the response with the solution of your problem. You will also find many new ideas regarding python technology and change in the versions.

(4) Python is Designed For Better Code Readability

Python provides a much better code readability as compared to another programming language. For example, it uses whitespace indentation in place of curly brackets for delimiting the block of codes. Isn't it awesome?

(5)Python Has a Vast Community Support

This is what makes python a favorable choice for development purposes. If you are having problems writing python a program, you can post directly to python community and will get the response with the solution of your problem. You will also find many new ideas regarding python technology and change in the versions.

(6)Python is Designed For Better Code Readability

Python provides a much better code readability as compared to another programming language. For example, it uses whitespace indentation in place of curly brackets for delimiting the block of codes.

(7)Python Contains Fewer Lines Of Codes

Codes are written in python programming language complete in fewer lines thus reducing the efforts of programmers. Let's have a look on the following "Hello World" program written in C, C++, Java, and Python.

While, C, C++, and Java take six, seven and five lines respectively for a simple "Hello World" program. Python takes only a single line which means, less coding effort and time is required for writing the same program.

Future Technologies Counting On Python

Generally, we have seen that python programming language is extensively used for web development, application development, system administration, developing games etc.

But do you know there are some future technologies that are relying on python? As a matter of fact, Python has become the core language as far as the success of these technologies is concerned. Let's dive into the technologies which use python as a core element for research, production and further developments.

(1) Artificial Intelligence (AI)

Python programming language is undoubtedly dominating the other languages when future technologies like Artificial Intelligence (AI) come into the play.

There are plenty of python frameworks, libraries, and tools that are specifically developed to direct Artificial Intelligence to reduce human efforts with increased accuracy and efficiency for various development purposes.

It is only the Artificial Intelligence that has made it possible to develop speech recognition system, autonomous cars, interpreting data like images, videos etc.

We have shown below some of the python libraries and tools used in various Artificial Intelligence branches.

- Machine Learning- PyML, PyBrain, scikit-learn, MDP Toolkit, GraphLab Create, MIPy etc.
- General AI- pyDatalog, AIMA, EasyAI, SimpleAI etc.
- Neural Networks- PyAnn, pyrenn, ffnet, neurolab etc.
- Natural Language & Text Processing- Quepy, NLTK, gensim

(2) Big Data

The future scope of python programming language can also be predicted by the way it has helped big data technology to grow. Python has been successfully contributing in analyzing a large number of data sets across computer clusters through its high- performance toolkits and libraries.

Let's have a look at the python libraries and toolkits used for Data analysis and handling other big data issues.

- Pandas
- Scikit-Learn
- NumPy
- SciPy
- GraphLab Create
- IPython
- Bokeh
- Agate
- PySpark
- Dask

(3)Networking

Networking is another field in which python has a brighter scope in the future. Python programming language is used to read, write and configure routers and switches and perform other networking automation tasks in a cost-effective and secure manner.

For these purposes, there are many libraries and tools that are built on the top of the python language. Here we have listed some of these python libraries and tools especially used by network engineers for network automation.

- Ansible
- Netmiko
- NAPALM(Network Automation and Programmability Abstraction Layer with Multivendor Support)
- Pyeapi
- Junos PyEZ
- PySNMP
- Paramiko SSH

Real-Life Python Success Stories

Python has seemingly contributed as a core language for increasing productivity regarding various development purposes at many of the IT organizations. We have shown below some of the real-life python success stories.

- Australia's RMA Department D-Link has successfully implemented python for creating DSL Firmware Recovery System.
- Python has helped Gusto.com, an online travel site, in reducing development costs and time.
- ForecastWatch.com also uses python in rating the accuracy of weather forecast reports provided by companies such as Accuweather, MyForecast.com and The Weather Channel.
- Python has also benefited many product development companies such as Acqutek, AstraZeneca, GravityZoo, Carmanah Technologies Inc. etc in creating autonomous devices and software.
- Test&Go uses python scripts for Data Validation.
- Industrial Light & Magic(ILM) also uses python for batch processing that includes modeling, rendering and compositing thousands of picture frames per day.

There is a huge list of success stories of many organizations across the globe which are using python for various purposes such as software development, data mining, unit testing, product development, web development, data validation, data visualization etc.

These success stories directly point towards a promising future scope of python programming language.

3.3Top Competitors Of Python

The future scope of python programming language also depends on its competitors in the IT market. But, due to the fact that it has become a core language for future technologies such as artificial intelligence, big data, etc., it will surely rise further and will be able to beat its competitors.

Competitors and Alternatives to Python Programming Language

- Microsoft.
- Oracle.
- IBM.
- Tableau.
- SAP.
- Alteryx.
- Blue Yonder.
- Gurobi.

4. Hardware and Software Requirements

Hardware Requirements

- Speed: 233MHz and above

- Hard Disk 10 GB
- RAM: 256 MB

Software Requirements

- Operating System: Windows/Linux Front End: Python 3.7
- Platform: Anaconda
- RAM: 256 MB

5. Formal description of the Training Models used

This section aims to formally define the various training models used in this study.

5.1 Naïve Bayes model - Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naïve Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naïve Bayes :

- Real time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- Multi class Prediction: This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- Recommendation System: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

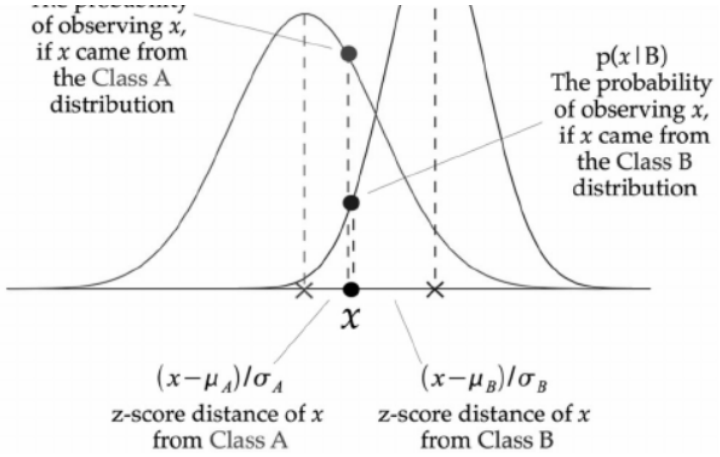
Different Models of Naïve Bayes used –

- Gaussian Naïve Bayes - is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. The likelihood of the features is assumed to be-

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.



The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently.

- Multinomial Naïve Bayes - Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

- Bernoulli Naïve Bayes - This is used for discrete data and it works on Bernoulli distribution. The main feature of Bernoulli Naive Bayes is that it accepts features only as binary values like true or false, yes or no, success or failure, 0 or 1 and so on. So when the feature values are binary we know that we have to use Bernoulli Naive Bayes classifier.

The Bernoulli distribution

$$p(x) = P[X = x] = \begin{cases} q = 1 - p & x = 0 \\ p & x = 1 \end{cases}$$

Bernoulli Naive Bayes Classifier is based on the following rule:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

5.2 Artificial Neural Network (ANN)

An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available.

- An artificial neural network (ANN) is the component of artificial intelligence that is meant to simulate the functioning of a human brain.
- Processing units make up ANNs, which in turn consist of inputs and outputs. The inputs are what the ANN learns from to produce the desired output.
- Back propagation is the set of learning rules used to guide artificial neural networks.

- The practical applications for ANNs are far and wide, encompassing finance, personal communication, industry, education, and so on

Artificial neural networks are built like the human brain, with neuron nodes interconnected like a web. The human brain has hundreds of billions of cells called neurons. Each neuron is made up of a cell body that is responsible for processing information by carrying information towards (inputs) and away (outputs) from the brain.

An ANN has hundreds or thousands of artificial neurons called processing units, which are interconnected by nodes. These processing units are made up of input and output units. The input units receive various forms and structures of information based on an internal weighting system, and the neural network attempts to learn about the information presented to produce one output report. Just like humans need rules and guidelines to come up with a result or output, ANNs also use a set of learning rules called back propagation, an abbreviation for backward propagation of error, to perfect their output results.

An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired output. The difference between both outcomes is adjusted using back propagation. This means that the network works backward, going from the output unit to the input units to adjust the weight of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error.

Practical Applications for Artificial Neural Networks (ANNs)

Artificial neural networks are paving the way for life-changing applications to be developed for use in all sectors of the economy. Artificial intelligence platforms that are built on ANNs are disrupting the traditional ways of doing things. From translating web pages into other languages to having a virtual assistant order groceries online to conversing with chat bots to solve problems, AI platforms are simplifying transactions and making services accessible to all at negligible costs.

Artificial neural networks have been applied in all areas of operations. Email service providers use ANNs to detect and delete spam from a user's inbox; asset managers use it to forecast the direction of a company's stock; credit rating firms use it to improve their credit scoring methods; e-commerce platforms use it to personalize recommendations to their audience; chat bots are developed with ANNs for natural language processing; deep learning algorithms use ANN to predict the likelihood of an event; and the list of ANN incorporation goes on across multiple sectors, industries, and countries.

In a neural network, there are three essential layers –

Input Layers

The input layer is the first layer of an ANN that receives the input information in the form of various texts, numbers, audio files, image pixels, etc.

Hidden Layers

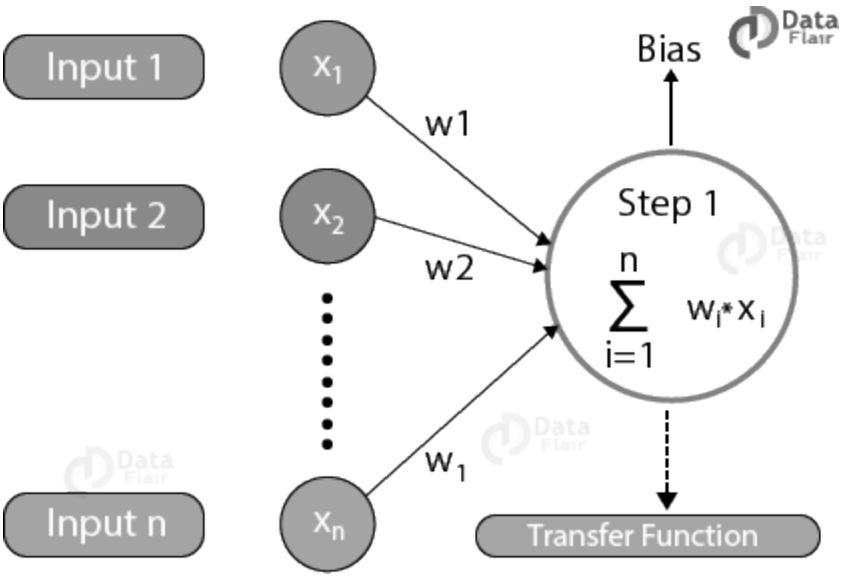
In the middle of the ANN model are the hidden layers. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These hidden layers perform various types of mathematical computation on the input data and recognize the patterns that are part of.

Output Layer

In the output layer, we obtain the result that we obtain through rigorous computations performed by the middle layer.

In a neural network, there are multiple parameters and hyperparameters that affect the performance of the model. The output of ANNs is mostly dependent on these parameters. Some of these parameters are weights, biases, learning rate, batch size etc. Each node in the ANN has some weight.

Each node in the network has some weights assigned to it. A transfer function is used for calculating the weighted sum of the inputs and the bias. After the transfer function has calculated the sum, the activation function obtains the result. Based on the output received, the activation functions fire the appropriate result from the node. For example, if the output received is above 0.5, the activation function fires a 1 otherwise it remains 0.



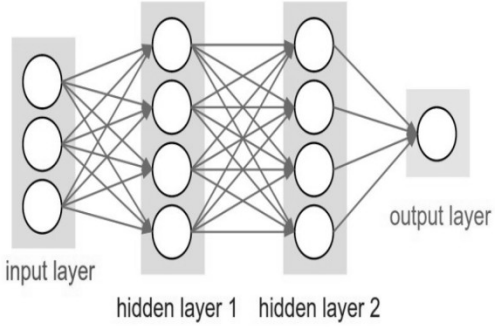
5.3 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/**CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

One of the main parts of Neural Networks is Convolutional neural networks (CNN). CNNs use image recognition and classification in order to detect objects, recognize faces, etc. They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output.

The first layer in a CNN network is the CONVOLUTIONAL LAYER, which is the core building block and does most of the computational heavy lifting. Data or imaged is convolved using filters or kernels.



5.4 Recurrent Neural Network

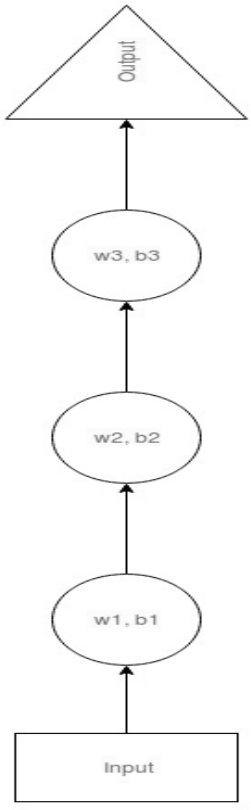
Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step**.** In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a **"memory"** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.

- Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.

Suppose there is a deeper network with one input layer, three hidden layers and one output layer. Then like other neural networks, each hidden layer will have its own set of weights and biases, let's say, for hidden layer 1 the weights and biases are (w1, b1), (w2, b2) for second hidden layer and (w3, b3) for third hidden layer. This means that each of these layers are independent of each other, i.e. they do not memorize the previous outputs.



Now the RNN will do the following:

- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.
- Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.

Training through RNN

1. A single time step of the input is provided to the network.
2. Then calculate its current state using set of current input and the previous state.
3. The current ht becomes ht-1 for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

5.4.1 LSTMs

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

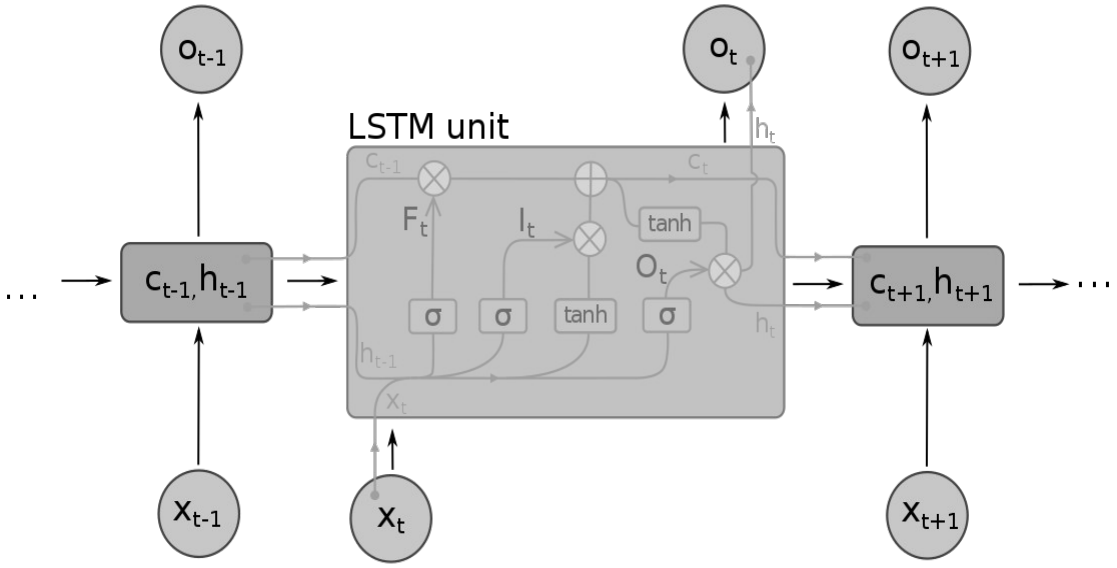
The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a point wise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.



6. Optimizers

Optimizers are algorithms or methods used to change the attributes of the **neural network** such as weights and learning rate to reduce the losses. **Optimizers** are used to solve optimization problems by minimizing the function.

Gradient Descent

Gradient Descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms. Backpropagation in neural networks also uses a gradient descent algorithm.

Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.

Advantages:

1. Easy computation.
2. Easy to implement.
3. Easy to understand.

Disadvantages:

1. May trap at local minima.
2. Weights are changed after calculating gradient on the whole dataset. So, if the dataset is too large than this may take years to converge to the minima.
3. Requires large memory to calculate gradient on the whole dataset.

Stochastic Gradient Descent

It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in Gradient Descent.

As the model parameters are frequently updated parameters have high variance and fluctuations in loss functions at different intensities.

Advantages:

1. Frequent updates of model parameters hence, converges in less time.
2. Requires less memory as no need to store values of loss functions.
3. May get new minima's.

Disadvantages:

1. High variance in model parameters.
2. May shoot even after achieving global minima.
3. To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.

Mini-Batch Gradient Descent

It's best among all the variations of gradient descent algorithms. It is an improvement on both SGD and standard gradient descent. It updates the model parameters after every batch. So, the dataset is divided into various batches and after every batch, the parameters are updated.

Advantages:

1. Frequently updates the model parameters and also has less variance.
2. Requires medium amount of memory.

All types of Gradient Descent have some challenges:

1. Choosing an optimum value of the learning rate. If the learning rate is too small than gradient descent may take ages to converge.
2. Have a constant learning rate for all the parameters. There may be some parameters which we may not want to change at the same rate.
3. May get trapped at local minima.

Momentum

Momentum was invented for reducing high variance in SGD and softens the convergence. It accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction. One more hyperparameter is used in this method known as momentum symbolized by 'γ'.

$$V(t)=\gamma V(t-1)+\alpha.\nabla J(\theta)$$

Now, the weights are updated by $\theta=\theta-V(t)$.

The momentum term γ is usually set to 0.9 or a similar value.

Advantages:

1. Reduces the oscillations and high variance of the parameters.
2. Converges faster than gradient descent.

Disadvantages:

1. One more hyper-parameter is added which needs to be selected manually and accurately.

Nesterov Accelerated Gradient

Momentum may be a good method but if the momentum is too high the algorithm may miss the local minima and may continue to rise up. So, to resolve this issue the NAG algorithm was developed. It is a look ahead method. We know we'll be using $\gamma V(t-1)$ for modifying the weights so, $\theta - \gamma V(t-1)$ approximately tells us the future location. Now, we'll calculate the cost based on this future parameter rather than the current one.

$V(t) = \gamma V(t-1) + \alpha . \nabla J(\theta - \gamma V(t-1))$ and then update the parameters using $\theta = \theta - V(t)$.

Adagrad

One of the disadvantages of all the optimizers explained is that the learning rate is constant for all parameters and for each cycle. This optimizer changes the learning rate. It changes the learning rate **'η'** for each parameter and at every time step **'t'.** It's a type second order optimization algorithm. It works on the derivative of an error function.

Advantages:

1. Learning rate changes for each training parameter.
2. Don't need to manually tune the learning rate.
3. Able to train on sparse data.

Disadvantages:

1. Computationally expensive as a need to calculate the second order derivative.
2. The learning rate is always decreasing results in slow training.

AdaDelta

It is an extension of **AdaGrad** which tends to remove the *decaying learning Rate* problem of it. Instead of accumulating all previously squared gradients, **Adadelta** limits the window of accumulated past gradients to some fixed size **w**. In this exponentially moving average is used rather than the sum of all the gradients.

Advantages:

1. Now the learning rate does not decay and the training does not stop.

Disadvantages:

1. Computationally expensive.

Adam

Adam (Adaptive Moment Estimation) works with momentums of first and second order. The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search.

Advantages:

1. The method is too fast and converges rapidly.
2. Rectifies vanishing learning rate, high variance.

Disadvantages:

1. Computationally costly.

7. Loss Functions

Loss functions play an important role in any statistical model - they define an objective which the performance of the model is evaluated against and the parameters learned by the model are determined by minimizing a chosen loss function.

Loss functions define what a good prediction is and isn't. In short, choosing the right loss function dictates how well your estimator will be. This article will probe into loss functions, the role they play in validating predictions, and the various loss functions used.

Loss functions for classification

Classification problems involve predicting a discrete class output. It involves dividing the dataset into different and unique classes based on different parameters so that a new and unseen record can be put into one of the classes.

A mail can be classified as a spam or not a spam and a person's dietary preferences can be put in one of three categories - vegetarian, non-vegetarian and vegan. Let's take a look at loss functions that can be used for classification problems.

Binary Cross Entropy Loss

This is the most common loss function used for classification problems that have two classes. The word "entropy", seemingly out-of-place, has a statistical interpretation.

Entropy is the measure of randomness in the information being processed, and cross entropy is a measure of the difference of the randomness between two random variables.

If the divergence of the predicted probability from the actual label increases, the cross-entropy loss increases. Going by this, predicting a probability of .011 when the actual observation label is 1 would result in a high loss value. In an ideal situation, a “perfect” model would have a log loss of 0.

Categorical Cross Entropy Loss

Categorical Cross Entropy loss is essentially Binary Cross Entropy Loss expanded to multiple classes. One requirement when categorical cross entropy loss function is used is that the labels should be one-hot encoded.

This way, only one element will be non-zero as other elements in the vector would be multiplied by zero.

Hinge Loss

Another commonly used loss function for classification is the hinge loss. Hinge loss is primarily developed for support vector machines for calculating the maximum margin from the hyperplane to the classes.

Loss functions penalize wrong predictions and does not do so for the right predictions. So, the score of the target label should be greater than the sum of all the incorrect labels by a margin of (at the least) one.

This margin is the maximum margin from the hyperplane to the data points, which is why hinge loss is preferred for SVMs.

8. Activation Functions

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

Activation functions are a critical part of the design of a neural network.

The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make.

Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.

A network may have three types of layers: input layers that take raw input from the domain, hidden layers that take input from another layer and pass output to another layer, and output layers that make a prediction.

All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model.

Most commonly used activation functions are as follows –

- Rectified Linear Activation (**ReLU**)
- Logistic (**Sigmoid**)
- Hyperbolic Tangent (**Tanh**)

ReLU Activation Function

The rectified linear activation function, or ReLU activation function, is perhaps the most common function used for hidden layers.

It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or "*dead*" units.

The ReLU function is calculated as follows:

- max(0.0, x)

This means that if the input value (x) is negative, then a value 0.0 is returned, otherwise, the value is returned.

Sigmoid Activation Function

The sigmoid activation function is also called the logistic function.

It is the same function used in the logistic regression classification algorithm.

The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0.

The sigmoid activation function is calculated as follows:

- 1.0 / (1.0 + e^-x)

Where e is a mathematical constant, which is the base of the natural logarithm.

Tanh Activation Function

The hyperbolic tangent activation function is also referred to simply as the Tanh (also "*tanh*" and "*TanH*") function.

It is very similar to the sigmoid activation function and even has the same S-shape.

The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

The Tanh activation function is calculated as follows:

- (e^x – e^-x) / (e^x + e^-x)

Where e is a mathematical constant that is the base of the natural logarithm.

9. Source Code Snippets

9.1 Data preprocessing

```
import numpy as np
import pandas as pd
import seaborn as sns

df=pd.read_csv('/IMDB Dataset.csv.zip')

df.head()

#data preprocessing

df=df.sample(10000)

#using one hot encoding convert 'positive' to 1 and 'negative' to 0

df = pd.get_dummies(df, columns=['sentiment'], drop_first=True)

df.head(5)

#cleaning the data

#remove the html tags

import re

def remove_tag(org):
  find = re.compile('<.*?>')
  cleantext = re.sub(find, '',org)
  return cleantext

df['review']=df['review'].apply(remove_tag)

df.head()

#tags have been removed
#next function to remove special characters

def remove_special_characters(org):
  cleantext=re.sub(r'\W+,. ', ' ', org)
  return cleantext

df['review']=df['review'].apply(remove_special_characters)

df.head()

#convert every character to lower case

def convert_to_lower(org):
  cleantext=org.lower()
  return cleantext

df['review']=df['review'].apply(convert_to_lower)



#every character has been converted to lowercase

def remove_punctuation(org):
  cleantext=re.sub(r'[^\w\s]', ' ',org)
  return cleantext

df['review']=df['review'].apply(remove_punctuation)

df.head()

pip install nltk

import nltk

from nltk.corpus import stopwords

```

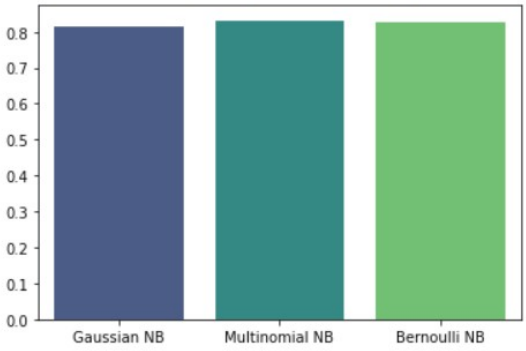
```
nltk.download('stopwords')

stopwords.words('english')

def remove_stopwords(org):
  cleantext=[]
  for word in org.split():
    if word not in stopwords.words('english'):
      cleantext.append(word)
  new=cleantext[:]
  cleantext.clear()
  return new

df['review']=df['review'].apply(remove_stopwords)

df.head()

from nltk.stem.porter import PorterStemmer

stemming_model=PorterStemmer()

def stem_words(org):
  cleantext=[]
  for word in org:
    cleantext.append(stemming_model.stem(word))
  new=cleantext[:]
  cleantext.clear()
  return new

df['review']=df['review'].apply(stem_words)

df.head()

def join_back(org):
  return " ".join(org)

df['review']=df['review'].apply(join_back)

df.head()

#converting updated dataframe to csv

df.to_csv('updated_sentiment_analysis.csv')

df2=pd.read_csv('updated_sentiment_analysis.csv')

df.head()

import warnings

warnings.filterwarnings('ignore', '.*do not.*', )

#end of preprocessing
 # converting to vectors from string
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=500)
x= cv.fit_transform(df2['review']).toarray()

print(x.shape)

y=df2.iloc[:,-1].values


#end of preprocessing
```

9.2 Using Naïve Bayes Models

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=2021)

print(x_train.shape)

from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB

gnb=GaussianNB()
mnb=MultinomialNB()
bnb=BernoulliNB()

gnb.fit(x_train,y_train)

mnb.fit(x_train,y_train)

bnb.fit(x_train,y_train)

y_gnb_pred=gnb.predict(x_test)

y_mnb_pred=mnb.predict(x_test)

y_bnb_pred=bnb.predict(x_test)

 from sklearn.metrics import accuracy_score

 print("The accuracy score of Gaussian Naive Bayes is: ", accuracy_score(y_test,y_gnb_pred))

 print("The accuracy score of Multinomial Naive Bayes is: ", accuracy_score(y_test,y_mnb_pred))

 print("The accuracy score of Bernoulli Naive Bayes is: ", accuracy_score(y_test,y_bnb_pred))

 #graphical comparison of accuracy

 import seaborn as sns

 sns.barplot(x=['Gaussian NB','Multinomial NB','Bernoulli NB'],
             y=[accuracy_score(y_test,y_gnb_pred),accuracy_score(y_test,y_mnb_pred),accuracy_score(y_test,y_bnb_pred)],palette='viridis')
```

Comparing the three accuracies by the different naïve Bayes model, the following graph was obtained.

```
[12] (14000, 500)
     The accuracy score of Gaussian Naive Bayes is:  0.8153333333333334
     The accuracy score of Multinomial Naive Bayes is:  0.8321666666666667
     The accuracy score of Bernoulli Naive Bayes is:  0.826
     <matplotlib.axes._subplots.AxesSubplot at 0x7fe7e2e65550>
```

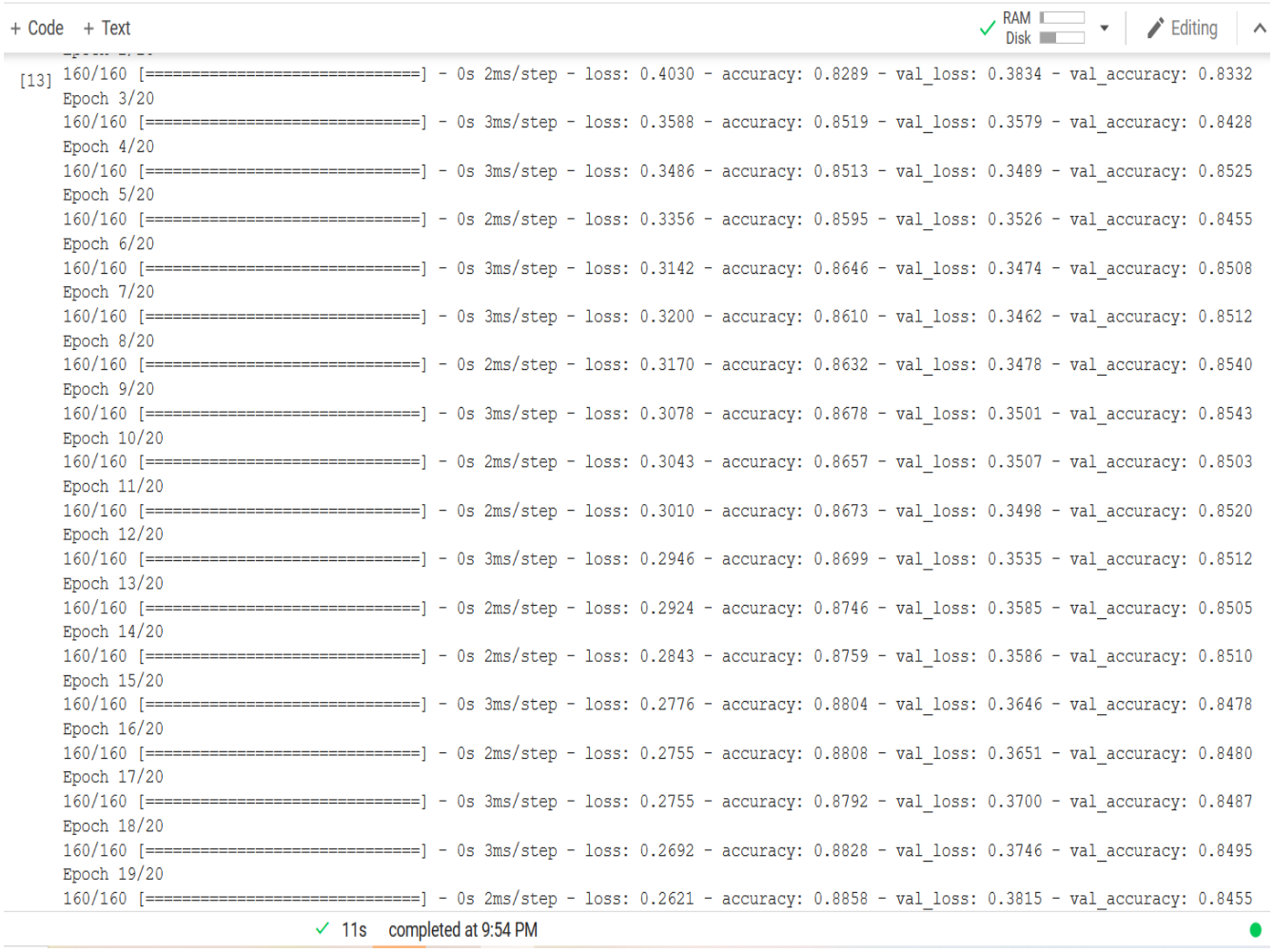


The inference that can be drawn is that all three models gave a very similar accuracy , but Multinomial Naïve Bayes performed marginally better than the rest.

9.3 Artificial Neural Network

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=500)
x= cv.fit_transform(df['review']).toarray()

y=df['sentiment_positive']

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=2021,test_size=0.2)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

ann_model=Sequential()

ann_model.add( Dense(8, activation="relu", input_shape = (x.shape[1],) ) )
ann_model.add( Dense(8, activation="relu") )
ann_model.add( Dense(1, activation="sigmoid"))

ann_model.compile(optimizer='adam', loss="binary_crossentropy", metrics=['accuracy'])

ann_model.fit(x_train,y_train, batch_size=100,epochs=20,verbose=1,validation_data=(x_test,y_test))
```

Accuracy Scores:



As seen in the above picture, the accuracy was nearly equal to 88% which is an improvement from the naïve bayes classifier.

9.4 Convolutional Neural Networks

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=2021,test_size=0.1)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Flatten

from keras.layers.embeddings import Embedding

from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

cnn_model=Sequential()

cnn_model.add(Embedding(1000, 300, input_length=max_words))
cnn_model.add(Conv1D(64,3,activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Conv1D(32,3,activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Dense(8, activation='relu'))
cnn_model.add(Dense(1, activation='sigmoid'))

cnn_model.compile(optimizer='adam', loss="binary_crossentropy", metrics=['accuracy'])

cnn_model.summary()

cnn_model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=50, batch_size=128, verbose=2)
```

Accuracy scores were noted as :

```
Epoch 23/50
141/141 - 18s - loss: 0.3896 - accuracy: 0.7760 - val_loss: 0.9093 - val_accuracy: 0.5923
Epoch 24/50
141/141 - 18s - loss: 0.3798 - accuracy: 0.7813 - val_loss: 0.9329 - val_accuracy: 0.5910
Epoch 25/50
141/141 - 18s - loss: 0.3734 - accuracy: 0.7830 - val_loss: 0.9616 - val_accuracy: 0.5918
Epoch 26/50
141/141 - 18s - loss: 0.3671 - accuracy: 0.7848 - val_loss: 1.0192 - val_accuracy: 0.5846
Epoch 27/50
141/141 - 18s - loss: 0.3635 - accuracy: 0.7881 - val_loss: 1.0028 - val_accuracy: 0.5927
Epoch 28/50
141/141 - 18s - loss: 0.3534 - accuracy: 0.7922 - val_loss: 1.0377 - val_accuracy: 0.5899
Epoch 29/50
141/141 - 18s - loss: 0.3467 - accuracy: 0.7961 - val_loss: 1.0716 - val_accuracy: 0.5885
Epoch 30/50
141/141 - 18s - loss: 0.3450 - accuracy: 0.7957 - val_loss: 1.0961 - val_accuracy: 0.5926
Epoch 31/50
141/141 - 18s - loss: 0.3392 - accuracy: 0.7989 - val_loss: 1.1242 - val_accuracy: 0.5836
Epoch 32/50
141/141 - 18s - loss: 0.3314 - accuracy: 0.8030 - val_loss: 1.1556 - val_accuracy: 0.5860
Epoch 33/50
141/141 - 18s - loss: 0.3269 - accuracy: 0.8049 - val_loss: 1.1842 - val_accuracy: 0.5886
Epoch 34/50
141/141 - 18s - loss: 0.3220 - accuracy: 0.8064 - val_loss: 1.2166 - val_accuracy: 0.5881
Epoch 35/50
141/141 - 18s - loss: 0.3157 - accuracy: 0.8090 - val_loss: 1.2617 - val_accuracy: 0.5852
Epoch 36/50
141/141 - 18s - loss: 0.3148 - accuracy: 0.8089 - val_loss: 1.2851 - val_accuracy: 0.5881
Epoch 37/50
141/141 - 18s - loss: 0.3114 - accuracy: 0.8109 - val_loss: 1.3136 - val_accuracy: 0.5827
Epoch 38/50
141/141 - 18s - loss: 0.3063 - accuracy: 0.8136 - val_loss: 1.3680 - val_accuracy: 0.5818
Epoch 39/50
141/141 - 18s - loss: 0.3035 - accuracy: 0.8147 - val_loss: 1.3873 - val_accuracy: 0.5810
Epoch 40/50
```

Hence , as seen in the above picture , the accuracy in CNN was nearly 80-81 %

9.5 Recurrent Neural Network

The syntax of implementation of RNN is given below .

```
import numpy as np
import pandas as pd
import seaborn as sns

df=pd.read_csv('/content/updated_sentiment_analysis(2).csv')

df.head()

df=df.sample(10000)
data=df

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=500, split=' ')
tokenizer.fit_on_texts(df['review'].values)
x = tokenizer.texts_to_sequences(df['review'])

x = pad_sequences(x, maxlen=80)
y = df['sentiment_positive']
```

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=2021,test_size=0.1)




from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, LSTM

rnn_model=Sequential()

rnn_model.add(Embedding(500, 64, input_length=80))

rnn_model.add(LSTM(192, dropout=0.2, recurrent_dropout=0.2, return_sequences = True))

rnn_model.add(LSTM(96, dropout=0.2, recurrent_dropout=0.2))

rnn_model.add(Dense(1, activation='sigmoid'))

rnn_model.compile(optimizer='adam', loss="binary_crossentropy", metrics=['accuracy'])

rnn_model.fit(
    x_train, y_train,
    validation_data = (x_test, y_test),
    epochs = 5,batch_size=32
)
```

The accuracy of RNN model is as shown below .

```
rnn_model.add(LSTM(192, dropout=0.2, recurrent_dropout=0.2, return_sequences = True))

rnn_model.add(LSTM(96, dropout=0.2, recurrent_dropout=0.2))

rnn_model.add(Dense(1, activation='sigmoid'))

rnn_model.compile(optimizer='adam', loss="binary_crossentropy", metrics=['accuracy'])

rnn_model.fit(
    x_train, y_train,
    validation_data = (x_test, y_test),
    epochs = 5,batch_size=32
)
```

```
Epoch 1/5
563/563 [==============================] - 298s 518ms/step - loss: 0.5076 - accuracy: 0.7294 - val_loss: 0.4239 - val_accuracy: 0.81
Epoch 2/5
563/563 [==============================] - 292s 519ms/step - loss: 0.3654 - accuracy: 0.8424 - val_loss: 0.3793 - val_accuracy: 0.83
Epoch 3/5
563/563 [==============================] - 293s 520ms/step - loss: 0.3488 - accuracy: 0.8486 - val_loss: 0.3805 - val_accuracy: 0.83
Epoch 4/5
563/563 [==============================] - 287s 509ms/step - loss: 0.3404 - accuracy: 0.8539 - val_loss: 0.3683 - val_accuracy: 0.83
Epoch 5/5
563/563 [==============================] - 286s 507ms/step - loss: 0.3261 - accuracy: 0.8582 - val_loss: 0.3554 - val_accuracy: 0.84
<tensorflow.python.keras.callbacks.History at 0x7fe79daad850>
```

As seen in the above picture , the accuracy shown by RNN model was nearly 85%

9. Results

- As seen above , among the four implementations , the ANN model gave the best result with an accuracy of over 88% , followed by RNN (85%) and CNN 1D (81).
- The naïve bayes models also had a similar accuracy as the CNN of about 81%

10. Conclusion

- Sentiment analysis was performed on the IMDB Dataset to classify movies as positive or negative according to user ratings.
- The various models used for this project are naïve bayes classifiers ,

  ANN ,CNN and RNN.

- Among the ones mentioned above , the most accurate results were given by the ANN model , followed by RNN .
- Thus it can be concluded that neural networks have performed better than the traditional machine learning models .

11.Citations and References

- Lakshmi Devi B., Varaswathi Bai V., Ramasubbareddy S., Govinda K. (2020) Sentiment Analysis on Movie Reviews. In: Venkata Krishna P., Obaidat M. (eds) Emerging Research in Data Engineering Systems and Computer Communications. Advances in Intelligent Systems and Computing, vol 1054. Springer, Singapore. https://doi.org/10.1007/978-981-15-0135-7_31

- https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

- https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/