

SPAMMING CLASSIFICATION IN NEURAL NET.

NAMRATHA L BEMANE

1NH16CS066

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Neural nets are extremely popular in the Machine Learning domain. A neural net is composed of multiple layers. It has an input layer in which you input the parameter x (the input of the program). The input is then passed through multiple hidden layers, finally getting one output at the final layer, called the output layer. An ANN is a collection of simple processing units which communicate with each other using a large number of weighted connections. Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. In the most general terms, a neural network is: “a system composed of many simple interconnected processing elements operating in parallel whose function is determined by network structure, connection strengths and the computation performed at the processing elements.”

The revival of interest in this approach to computing is partly due to the failure of conventional programming techniques, including rule-based AI, to solve some of the “hard” problems like machine vision, continuous speech recognition and machine learning. Each unit receives input from neighbour units or from external source and computes output which propagates to other neighbours. There is also a mechanism to adjust weights of the connections. Normally there are 3 types of units.

- **Input Unit:** which receives signal from external source.
- **Output Unit:** which sends data out of the network.
- **Hidden Unit:** which Receives and sends signals within the network.

Many of the units can work parallel in the system. ANN can be adapted to take a set of inputs and produce a desired set of outputs. This process is known as learning or training. There are two kinds of training in neural network.

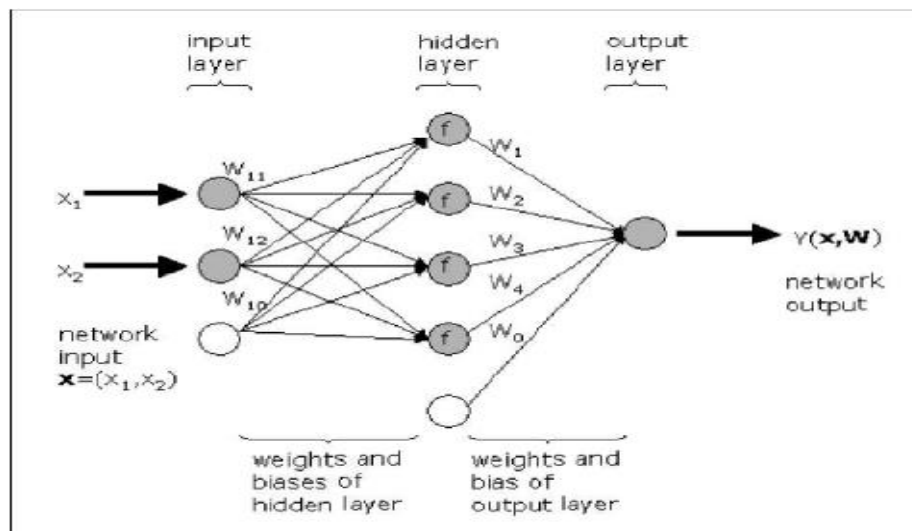
- **Supervised:** The network is provided a set of inputs and corresponding output patterns, called training dataset, to train the network.
- **Unsupervised:** The network trains itself by creating clusters of patterns. Here no prior set training data is provided to the system.

Multi-Layer Perceptron (MLP)

MLP is a nonlinear feed forward network model which maps a set of inputs x into a set of outputs y . It has 3 types of layers: input layer, output layer, and hidden layer. We have a very simple neural net, which consist of N hidden layers. Each layer contains one neuron. Each neuron has two values associated with it: w_i , and b_i , denoting the weight and the bias of the neuron. If you give the neuron an input of x , it produces an output of y . Standard perceptron calculates a discontinuous function:

$$y = w_1 * x + b_1 \text{ -----(1)}$$

smoothing is done using a logistic function to get the final output by running the logic for multiple times and classification of spammers from non-spammers is done. below figure (1) shows a sample neural net,



MLP is a finite acyclic graph where the nodes are neurons with logistic activation. Neurons of i layer serves as input for neurons of $(i+1)$ network containing many neurons. All the next layer.

Thus, an input x gets transformed by the neural net as follows. The first hidden neuron takes the input x and produces $y = w_1 * x + b_1$, which acts as the input for the second neuron. Then, the second neuron takes input y and produces an output of $z = w_2 * y + b_2$. This keeps happening and you get a single output at the end from the N th neuron i.e., $y(x, w)$

CHAPTER 2

ANALYSIS AND DESIGN

2.1 OBJECTIVES OF THE PROJECT

- To classify the given users in website or a platform into two categories i.e., spammers and non-spammers.
- The classification is done using their username and their user-id.
- To give a clear picture about the spamming classification using neural net format.
- Finally, getting the output in the format of number of spammer users and non-spammer users.

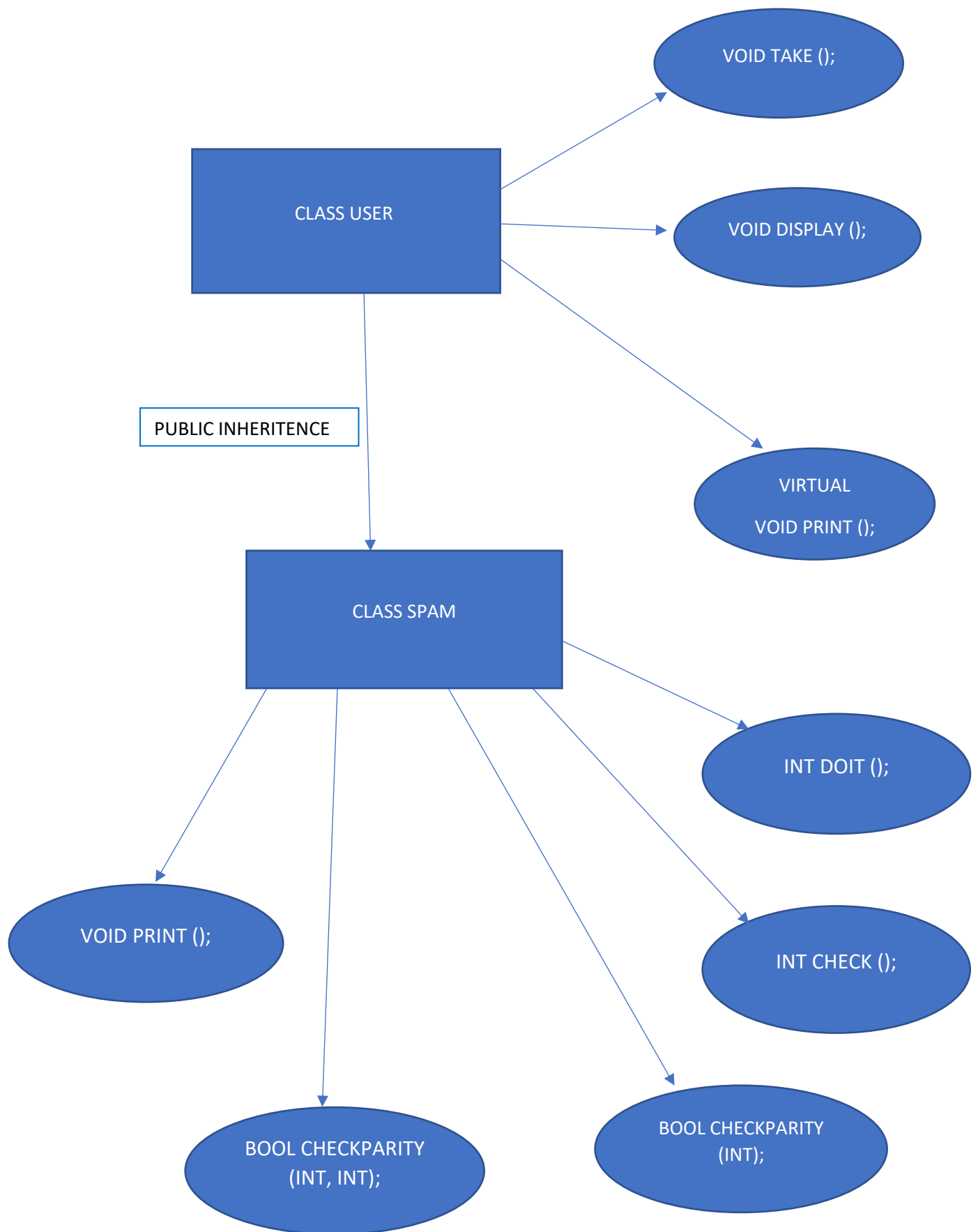
2.2 REQUIREMENT SPECIFICATION

- An IDE for running the object-oriented C++ code like DEV C++, TURBO C++.
- An C compiler for the built-in libraries i.e., MinGW compiler.
- A laptop with specifications like 2GB RAM, 4GB ROM, 12inch display, intel i3 processor etc.

2.3 ALGORITHM

- Create a class user with datatypes in protected mode and member function in public.
- Define a member function for taking the user name and user-id and another function for displaying the details taken.
- Create a function called print () and make it virtual.
- Inherit the base class user in a new derived class spam publicly.
- Define a member function doit () in class spam which takes the user's integer user-id in three space-separated integers **N**, **minx**, **maxX** format, and process them using the given function formula and counts the final result in two variables one for odd and other for even and stores them.
- Define a member function check () in spam class which the ASCII values of the username is taken and added and the result is stored in a variable.
- Define a member function called bool checkparity(int) which takes the result of check () and classifies them on the basis of being even for non-spammer or odd for spammer.
- Define another bool checkparity (int, int) which takes the result of doit () and classifies the users in odd and even basis for being a spammer or not.
- Create a void print () function same as base class and call this function through pointer.

2.4 CLASS DIAGRAM



CHAPTER 3

IMPLEMENTATION

3.1 ABSTRACTION

Data abstraction is one of the most essential and important feature of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

- **Abstraction using Classes:** We can implement Abstraction in C++ using classes. Class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to outside world and which is not.
- **Abstraction in Header files:** One more type of abstraction in C++ can be header files. For example, consider the `pow ()` method present in `math.h` header file. Whenever we need to calculate power of a number, we simply call the function `pow ()` present in the `math.h` header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating power of numbers.

3.2 ENCAPSULATION

Encapsulation is defined as wrapping up of data and information under a single unit. Encapsulation is defined as binding together the data and the functions that manipulates them. Encapsulation also lead to data abstraction or hiding. As using encapsulation also hides the data. The process of implementing encapsulation can be sub-divided into two steps:

- The data members should be labelled as private using the private access specifiers.
- The member function which manipulates the data members should be labelled as public using the public access specifier

3.3 CLASSES

The building block of C++ that leads to Object Oriented programming is a Class. It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object. A Class is a user defined data-type which have data members and member functions.

Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behaviour of the objects in a Class.

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

3.4 OBJECTS

An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Declaring Objects: When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Syntax: ClassName ObjectName;

CONSTRUCTORS: Constructors are special class members which are called by the compiler every time an object of that class is instantiated. Constructors have the same name as the class and may be defined inside or outside the class definition. There are 3 types of constructors:

- Default constructors
- Parametrized constructors
- Copy constructors

DESTRUCTORS: Destructor is another special member function that is called by the compiler when the scope of the object ends.

3.5 POLYMORPHISM

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Polymorphism is considered as one of the important features of Object Oriented Programming. In C++ polymorphism is mainly divided into two types:

- Compile time Polymorphism
- Runtime Polymorphism

- **Compile time polymorphism:** This type of polymorphism is achieved by function overloading or operator overloading.

Function Overloading: When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or change in the arguments below the rules for function overloading has been discussed.

Rules of Function Overloading: Function declarations that differ only in the return type. For example, the following program fails in compilation.

- Member function declarations with the same name and the name parameter-type-list cannot be overloaded if any of them is a static member function declaration. For example, following program fails in compilation.
- Parameter declarations that differ only in a pointer * versus an array [] are equivalent. That is, the array declaration is adjusted to become a pointer declaration. Only the second and subsequent array dimensions are significant in parameter types
- Parameter declarations that differ only in that one is a function type and the other is a pointer to the same function type are equivalent. For example, following two function declarations are equivalent.
- Parameter declarations that differ only in the presence or absence of const and/or volatile are equivalent. That is, the const and volatile type-specifiers for each parameter type are ignored when determining which function is being declared, defined, or called. For example, following program fails in compilation with error.
- Two parameter declarations that differ only in their default arguments are equivalent. For example, following program fails in compilation with error.

Operator Overloading: C++ also provide option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings. We know that this is the addition operator whose task is to add to operands. So, a single operator '+' when placed between integer operands, adds them and when placed between string operands, concatenates them.

- **Runtime polymorphism:** This type of polymorphism is achieved by Function Overriding. Function overriding on the other hand occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

3.6 INHERITENCE

The capability of a class to derive properties and characteristics from another class is called Inheritance.

- **Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.
- **Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

Syntax:

```
class subclass_name: access_mode base_class_name
```

```
{//body of subclass};
```

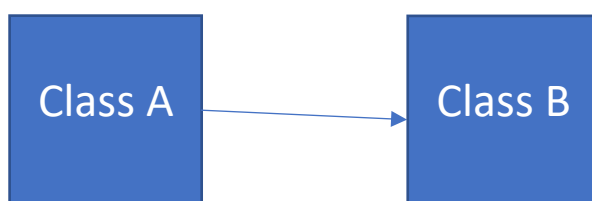
Here, subclass_name is the name of the sub class, access_mode is the mode in which you want to inherit this sub class for example: public, private etc. and base_class_name is the name of the base class from which you want to inherit the sub class. private member of the base class will never get inherited in the sub class.

➤ Modes of Inheritance

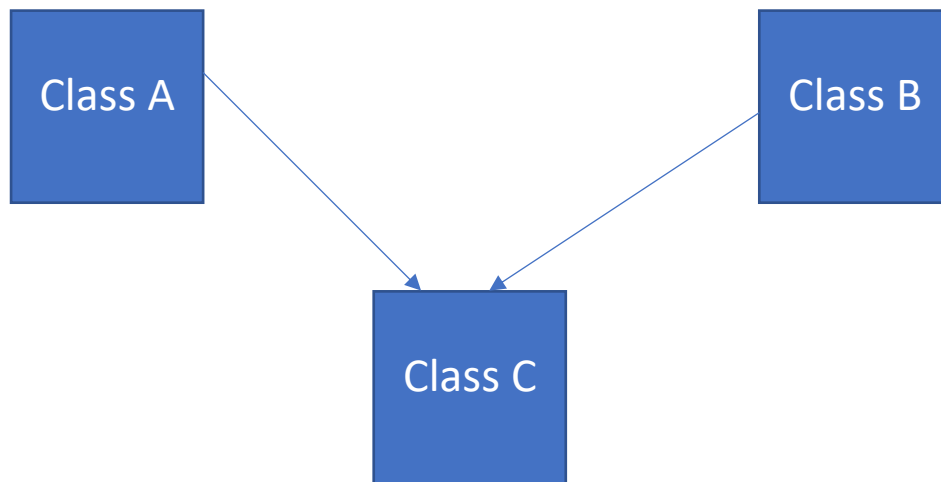
- **Public mode:** If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.
- **Protected mode:** If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.
- **Private mode:** If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class. Private members of the base class will never get inherited in sub class.

➤ Types of Inheritance

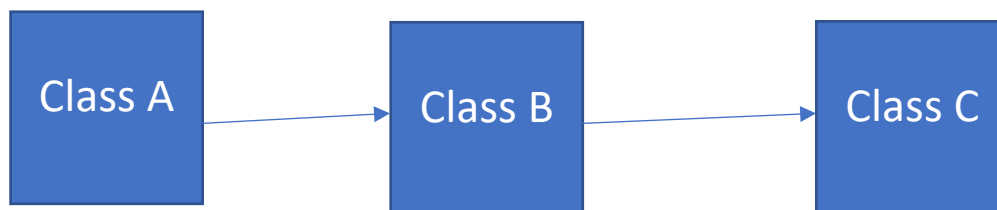
- **Single Inheritance:** In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.



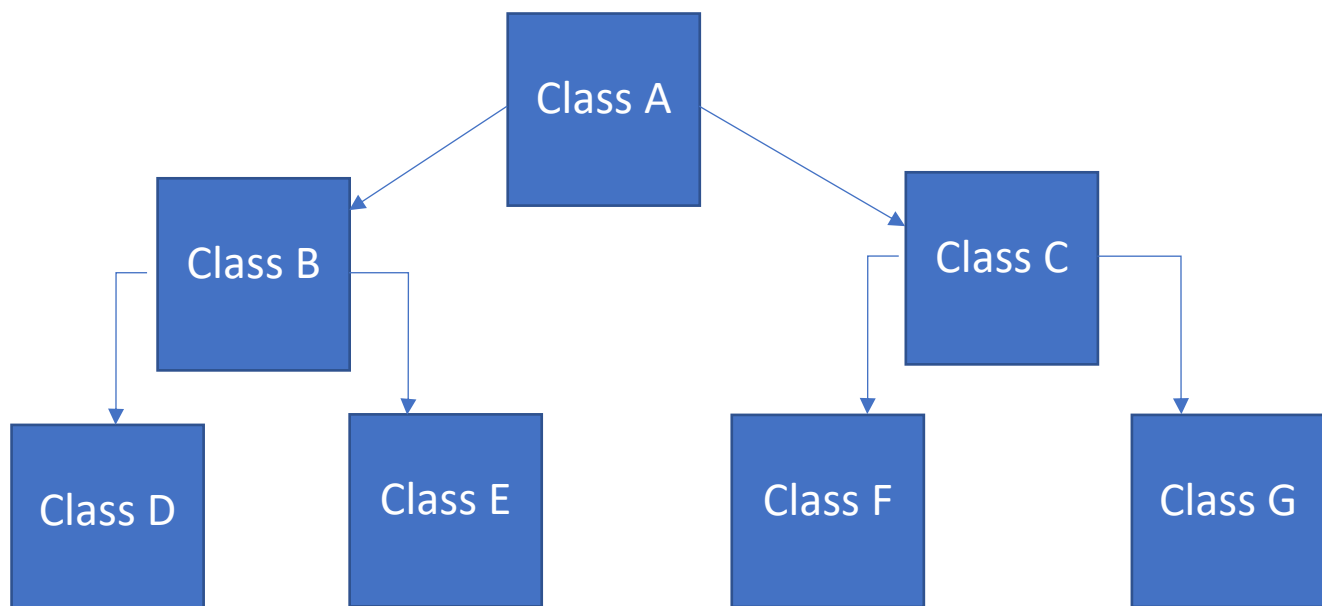
- **Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e. one **sub class** is inherited from more than one **base classes**.



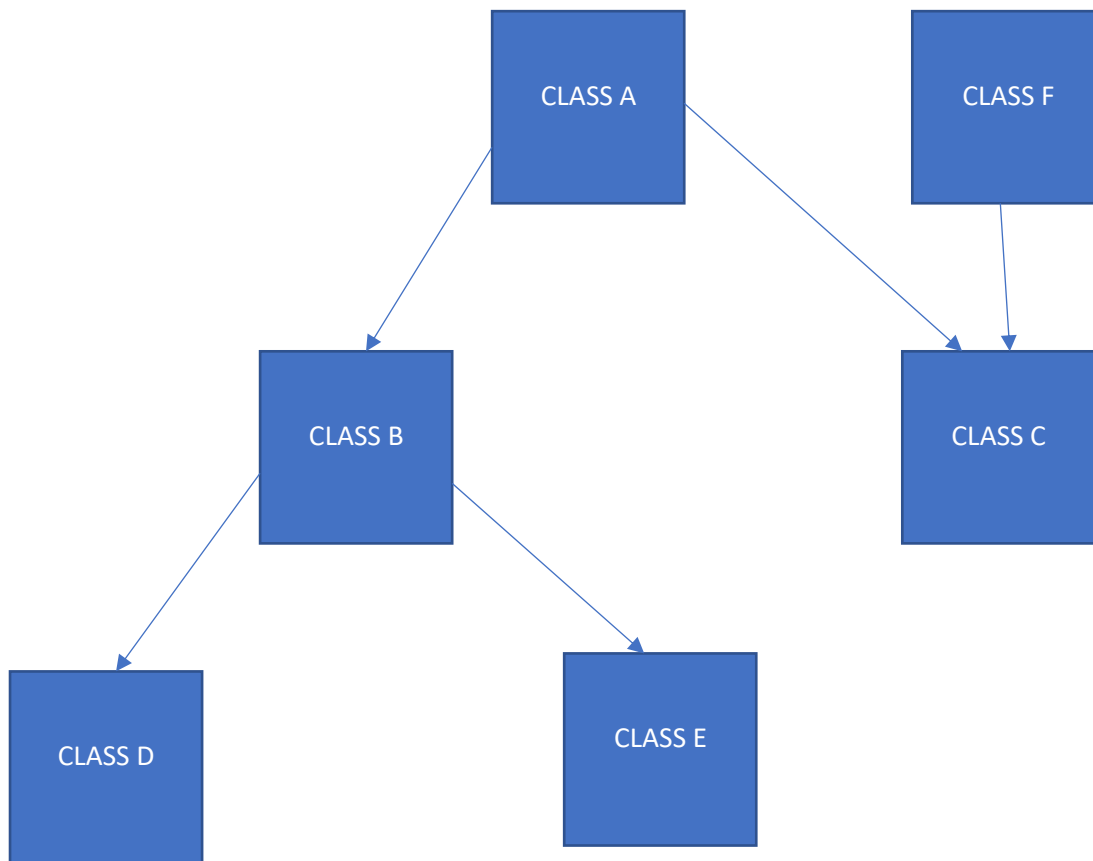
- **Multilevel Inheritance:** In this type of inheritance, a derived class is created from another derived class.



- **Hierarchical Inheritance:** In this type of inheritance, more than one sub class are inherited from a single base class. i.e. more than one derived class is created from a single base class.



- **Hybrid (Virtual) Inheritance:** Hybrid Inheritance is implemented by combining more than one type of inheritance. For-example: Combining-Hierarchical inheritance and Multiple-Inheritance. Below image shows the combination of hierarchical and multiple inheritance:



3.7 FRIEND FUNCTIONS

A friend function can be given special grant to access private and protected members. A friend function can be defined as

- A method of another class
- A global function

Following are the few points about friend functions and classes:

- 1) Friends should be used only for limited purpose. Too many functions or external classes are declared as friends of a class with protected or private data, it lessens the value of encapsulation of separate classes in object-oriented programming.
- 2) Friendship is not mutual. If a class A is friend of B, then B doesn't become friend of A automatically.
- 3) Friendship is not inherited.

3.8 VIRTUAL FUNCTIONS

A virtual function is a member function which is declared within a base class and is re-defined (Overridden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

- Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
- They are mainly used to achieve Runtime polymorphism
- Functions are declared with a virtual keyword in base class.
- The resolving of function call is done at Run-time.

3.9 DYNAMIC MEMORY ALLOCATION

Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer. C++ supports these functions and also has two operators `new` and `delete` that perform the task of allocating and freeing the memory in a better and easier way.

3.8.1 NEW OPERATOR

The `new` operator denotes a request for memory allocation on the Heap. If sufficient memory is available, `new` operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable. Syntax to use `new` operator:

To allocate memory of any data type, the **syntax is:** `pointer-variable = new data-type;`

3.8.2 DELETE OPERATOR

Since it is programmer's responsibility to deallocate dynamically allocated memory, programmers are provided with, `delete` operator in object-oriented C++ language to delete the memory occupied by objects in program.

Syntax: `delete pointer-variable;`

3.99 DYNAMIC BINDING


The compiler adds code that identifies the kind of object at runtime then matches the call with the right function definition. This can be achieved by declaring a virtual function. It occurs at run time also known as late binding. All information is needed at run time. It is flexible. It runs at slow rate

Example: - Virtual function in C++, overridden methods in java.

CHAPTER 4


SAMPLE OUTPUT

4.1 APPENDIX-1

 D:\mini project\a\trial\test.exe


```
*****OPTIONS*****  
1.CREATE LOGIN ID  
2.CHECK SPAM USING NAME  
3.CHECK SPAM USING USER-ID  
4.EXIT
```

4.2 APPENDIX -2


 D:\mini project\a\trial\test.exe

```
Enter username:  
NAMRATHALB  
USERNAME:NAMRATHALB  
  
do you want to continue the process???  
enter 1 continue 0 exit
```

4.3 APPENDIX-3

 D:\mini project\a\trial\test.exe


```
*****OPTIONS*****
1.CREATE LOGIN ID
2.CHECK SPAM USING NAME
3.CHECK SPAM USING USER-ID
4.EXIT
2_
```

 D:\mini project\a\trial\test.exe


```
USERNAME:NAMRATHALB
730 is the sum of USERNAME which is even and hence user is not a spammer

do you want to continue the process???
enter 1 continue 0 exit
```

4.4 APPENDIX-4

 D:\mini project\a\trial\test.exe

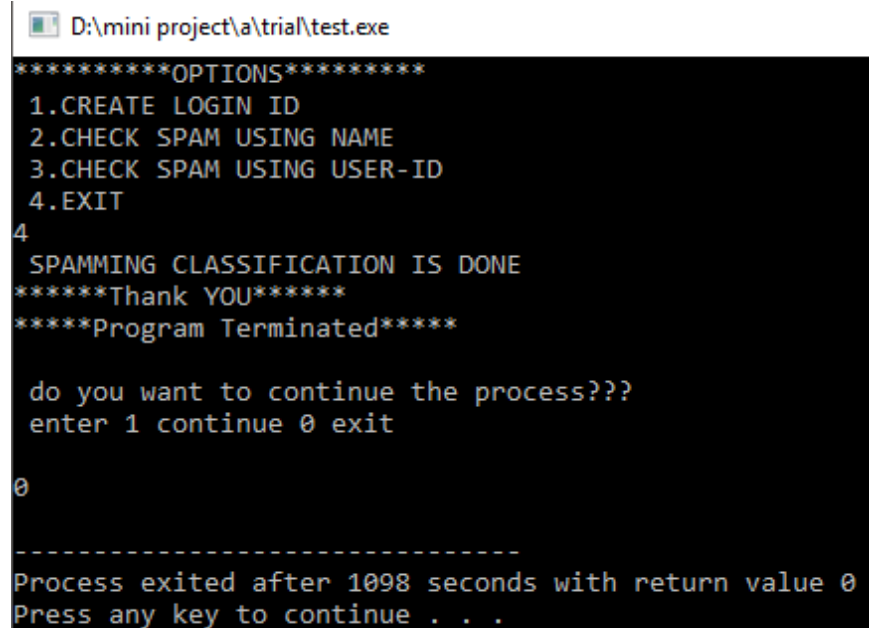
```
*****OPTIONS*****
1.CREATE LOGIN ID
2.CHECK SPAM USING NAME
3.CHECK SPAM USING USER-ID
4.EXIT
3_
```

 D:\mini project\a\trial\test.exe

```
USERNAME:NAMRATHALB
Enter the number of layers:
3
Enter userid
2 1 4
Enter the wieght and bias of the layer:
2 4
Enter the wieght and bias of the layer:
2 3
0 4
0 4 is the OUTPUT of NEURAL LAYER
There are 0 Non-spammer and 4 of spammers

do you want to continue the process???
enter 1 continue 0 exit
```

4.5 APPENDIX-5



```
D:\mini project\trial\test.exe
*****OPTIONS*****
1.CREATE LOGIN ID
2.CHECK SPAM USING NAME
3.CHECK SPAM USING USER-ID
4.EXIT
4
SPAMMING CLASSIFICATION IS DONE
*****Thank YOU*****
*****Program Terminated*****

do you want to continue the process???
enter 1 continue 0 exit

0

-----
Process exited after 1098 seconds with return value 0
Press any key to continue . . .
```

CHAPTER 5

CONCLUSION

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore, there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture. Neural networks also contribute to other areas of research such as neurology and psychology. Perhaps the most exciting aspect of neural networks is the possibility that someday 'conscious' networks might be produced. There is a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' neural networks are a realistic possibility.

The aim of this project is to demonstrate the high performance of a simple statistical filtering method: spamming classification. To fight against spam is a difficult problem for the entire world. In this paper we present a technique to classify the spammers from non-spammers using both username and user-id using artificial Neural Network with bias and weight measures. We have provided a simple framework for efficient spam filtering by using the concepts of object-oriented programming in C++. We also developed a simple algorithm for this classification which takes usernames and user-ids and feed it as input to the first layer of the neural net. Finally, we classify them as spammer or not on the basis of some condition on final output. We also have count the number of non-spammers and spammers and printed it at the end.

In this paper we review some of the most popular machine learning methods and of their applicability to the problem of spam classification of users in a website. If the number of factors is high, increased number of neurons improves the performance of algorithm. The training time increases for all algorithms when either number of inputs are increased or number of neurons in hidden layer are increased. For a final verdict about the algorithm's applicability to spam classification future research on larger spam data sets that are more representative of real life email or id traffic is needed.

Finally, this shows that fine tuning of network configuration and parameters is quite important in neural network research. Spam filtering is a race between spammers and spam filtering programmers, study and research in this field will never stop as long as the spam still appears in our email inbox. I would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.