# **Neural Networking Project Final Data Structure**

# Implementation:

#include<iostream>

#include<string>

#include<fstream>

#include<random>

#include<cmath>

#include<iomanip>

#include<ctime>

#include<windows.h>

using namespace std;

int ch;

class Neural\_Network {

private:

float Matrix[7][10];

int value = 0, bias = 0;

float sum = 0;

float sigmoid = 0;

int activation = 0, predicition = 0;

int Line = 0;

public:

void neuron(int data) {

ifstream file("tae.data");

for (int i = 1; i <= data; i++) {

for (int i = 1; i <= 6; i++) {

file >> value;

if (i == 1) {

bias = value;

}

else if (value != '\n') {

Matrix[i - 1][1] = value;

}

}

Line = i;

auto\_generate();

}

file.close();

}

void auto\_generate() {

srand(time(NULL));

float fMin = -5.0;

float fMax = 5.0;

for (int i = 1; i <= 5; i++) {

float f = (float)rand() / RAND\_MAX;

float x = fMin + f \* (fMax - fMin);

Matrix[i][2] = x;

}

Connection();

}

void Connection() {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= 2; j++) {

Matrix[i][3] = (Matrix[i][1]) \* (Matrix[i][2]);

}

}

Sum();

}

void Sum() {

for (int i = 1; i <= 5; i++) {

sum = sum + Matrix[i][3];

}

Sigmoid();

Activation();

if (ch == 1) {

Print();

}

else if (ch == 2) {

cout << setw(50) << "Line " << Line << " Data\n";

print\_Matrix();

}

}

void Sigmoid() {

sigmoid = (1 / (1 + exp(-(sum))));

if (sigmoid >= 0.3) {

sigmoid = 1;

}

else {

sigmoid = 0;

}

}

void Activation() {

activation = sum + bias;

if (activation >= 0.00) {

predicition = 1.0;

}

else {

predicition = 0.00;

}

}

void Print() {

cout << endl;

for (int i = 1; i <= 5; i++) {

cout <<" " << Matrix[i][1] << " ";

}

cout << "\t\t\t\t" << bias << "\t\t" << sigmoid << "\t\t" << predicition;

}

void percepton(int data) {

if (ch == 1) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 10);

cout << "\n The Given values of X\t\t" << " \tBias value\t" << " Output value\t" << "predicted values\n";

neuron(data);

}

if (ch == 2) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 4);

cout << "\nGiven values\t\t\t Weight\t\t\t\tMultiplication\n\n";

neuron(data);

}

}

void print\_Matrix() {

for (int i = 1; i <= 5; i++) {

for (int j = 1; j <= 3 ; j++) {

cout<<" " << Matrix[i][j] << "\t\t\t\t";

}

cout << endl;

}

cout << "\n\tBias =" << bias << "\n\tOutput =" << sigmoid << "\n\tPredicted =" << predicition;

cout << endl;

}

};

int main() {

Neural\_Network obj;

int data;

bool f=true;

do {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 4);

cout <<"\n" << setw(100) << " Welcome to Project of Neural Networking\n\n";

SetConsoleTextAttribute(hConsole, 11);

cout << "Enter 1 for print output\nEnter 2 for print procedure\n";

cin >> ch;

cout << "Enter How many data you want to retrive : Max 151\n";

cin >> data;

SetConsoleTextAttribute(hConsole, 2);

cout << "\t\tData is retriving\n";

Sleep(3000);

SetConsoleTextAttribute(hConsole, 9);

cout << "\t\twait ........\n";

Sleep(3000);

SetConsoleTextAttribute(hConsole, 10);

cout << "\t\tCompiler is start learning...\n";

Sleep(3000);

SetConsoleTextAttribute(hConsole, 9);

cout << "\t\tEnter any key for Output\n";

cout<<"\t\t"<<system("pause");

obj.percepton(data);

cout << endl;

cout<<endl<<"\t\t"<<system("pause");

system("cls");

} while (f = true);

}

# Output:

Text

Description automatically generated

Text

Description automatically generated