Felhasználói dokumentáció

- a program elindításakor két menüpont jelenik meg 1. tanitas, ha ezt a menüt szeretné választani akkor az 1-es számot kell beírni ha a 2. kep beolvasasa menüt szeretné választani akkor a 2-es számot kell megadni

  1. menüpont (tanitas):
     a hálózat ki és bemeneti rétegéi fix.
     - először meg kell adni hogy hány réteget szeretnénk (a ki és bemeneti rétegeken kívül)
     - utána meg kell adni a rétegek bemenetei számát szóközzel elválasztva (pl. 3-réteg esetén: 50 30 45)
     - utána meg kell adni a minta méretét tehát hány képpel szeretnénk tanítani
     - epoch száma: meg kell adni, hogy hányszor fusson le a tanítási algoritmus
     - és meg kell adni a tanulási rátát, amit érdemes 0.01 és 1-között megadni
     - miután véget ért a tanítás a program megkérdezi, hogy milyen néven szeretné menteni a hálózatot, amit .dat kiterjesztéssel, vagy kiterjesztés nélkül lehet megadni

2. menü pont (kep beolvasasa)
    - először meg kell adni az elmentett hálózat nevét, amin le akarja futtatni a képeket (ezt lehet kiterjesztés nélkül vagy .dat kiterjesztéssel)
    - ez után meg kell adni a (28x28-as pixelű) kép nevét és elérési útvonalát (a képen szereplő számnak fekete háttere kell legyen és a szám színe pedig fehér kell legyen küllőmben nem hamis eredményt fog adni) (itt muszáj megadni a kép kiterjesztését is)
        i. ha a kép beolvasása sikeres volt akkor kirajzolja a képet és kiírja, hogy milyen szám szerepel a képen
    - a program végén megkérdezi, hogy tovább szeretné-e folytatni vagy nem, ha igen y-t kell megadni, ha nem akkor n-t kell beírni
        i. ha y-t irt be akkor nem kell újra betölteni a hálózatot a program automatikusan elmenti az első használat után

Neural Network

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Layer Class Reference

Public Member Functions

- **Layer** (int inputs=1, int outputs=1)
- Matrix< double > **getW** () const
- Matrix< double > **getB** () const
- void **setW** (Matrix< double > w)
- void **setB** (Matrix< double > b)
- Matrix< double > forward (const Matrix< double > &input)

  ez a fügvény kiszámolja az adott réteg kimeneti értékeit
- Matrix< double > backprop (const Matrix< double > &error, const Matrix< double > &prev_out, const Matrix< double > &z, double lr)

  ezzel a fügvénnyel lehet tanítani a hálozatott

3.1.1 Member Function Documentation

3.1.1.1 backprop()

```
Matrix< double > Layer::backprop (
            const Matrix< double > & error,
            const Matrix< double > & prev_out,
            const Matrix< double > & z,
            double lr )
```

ezzel a fügvénnyel lehet tanítani a hálozatott

Parameters

error	az l+1. réteg hibája (deriváltja)
prev_out	az l-1. réteg kimenete (aktivációs fügvény után)
z	az l-1 réteg kimenete aktivációs fügvény meghivása elött
lr	tanulási ráta

Returns

l-1. réteg hibája

3.1.1.2 forward()

```
Matrix< double > Layer::forward (
            const Matrix< double > & input )
```

ez a fügvény kiszámolja az adott réteg kimeneti értékeit

Parameters

input	az adat vagy az előző réteg neuron rétékei

Returns

a réteg kimeneti neuronjai mátrixban eltárolva

The documentation for this class was generated from the following files:

- layer.h
- layer.cpp

3.2 Matrix< T > Class Template Reference

Public Member Functions

- Matrix (int row=1, int col=1)
  matrix konstruktora
- Matrix (const Matrix< T > &rhs)
  Másoló konstruktor.
- ~**Matrix** ()
  a Matrix Destruktora amely felszabadítja a dinamikusan foglalt 2D-tömböt
- int getRow () const
  lekérdezi a matrix hosszát
- int getCol () const
  lekérdezi a matrix szélességét
- T & operator() (int r, int c)
  ezzel a fügvénnyel lehet elérni a matrix egyes elemeit
- const T **operator()** (int r, int c) const
- Matrix< T > operator+ (const Matrix< T > &rhs) const
  két azonos méretű mátrixot add össze
- Matrix< T > operator* (const Matrix< T > &rhs) const
  Hadamard-szorzat, amely elementként szorozza össze a mátrixokat.
- Matrix< T > & **operator=** (const Matrix< T > &rhs)
- Matrix< T > operator- (const Matrix< T > &rhs)
  mátrix kivonás
- void **mrand** ()
  a mátrixot -0.5 és 0.5 közötti véletlen számokkal tölti fel
- void **zeros** ()
  a mátrix minden elemét nullával tölti fel
- Matrix< T > Transpose () const
  ez a fügvény a mátrix trasznponáltjával tér vissza

Friends

- std::ostream & operator<< (std::ostream &os, const Matrix< T > &m)

  kiíró fugvény

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Matrix() [1/2]

```
template<typename T >
Matrix< T >::Matrix (
            int row = 1,
            int col = 1 )  [inline]
```

matrix konstruktora

Parameters

row	hosz
col	szélesség

3.2.1.2 Matrix() [2/2]

```
template<typename T >
Matrix< T >::Matrix (
            const Matrix< T > & rhs )  [inline]
```

Másoló konstruktor.

Parameters

rhs	másolando objektum

3.2.2 Member Function Documentation

3.2.2.1 getCol()

```
template<typename T >
int Matrix< T >::getCol ( ) const  [inline]
```

lekérdezi a matrix szélességét

Returns

mátrix szélessége

3.2.2.2 getRow()

```
template<typename T >
int Matrix< T >::getRow ( ) const  [inline]
```

lekérdezi a matrix hosszát

Returns

matrix hossza

3.2.2.3 operator()()

```
template<typename T >
T & Matrix< T >::operator() (
            int r,
            int c )  [inline]
```

ezzel a fügvénnyel lehet elérni a matrix egyes elemeit

Parameters

r	sorszám
c	oszlopszám

Returns

a matrix r. sor c. oszlop eleme

3.2.2.4 operator∗()

```
template<typename T >
Matrix< T > Matrix< T >::operator* (
            const Matrix< T > & rhs ) const  [inline]
```

Hadamard-szorzat, amely elementként szorozza össze a mátrixokat.

Parameters

rhs	jobb oldali mátrix

Returns

eredmény mátrix

3.2.2.5 operator+()

```
template<typename T >
```

```
Matrix< T > Matrix< T >::operator+ (
            const Matrix< T > & rhs ) const  [inline]
```

két azonos méretű mátrixot add össze

Parameters

rhs	jobb oldali mátrix

Returns

összeg mátrix

3.2.2.6 operator-()

```
template<typename T >
Matrix< T > Matrix< T >::operator- (
            const Matrix< T > & rhs )  [inline]
```

mátrix kivonás

Parameters

rhs	jobb oldali mátrix

Returns

eredmény mátrix

3.2.2.7 Transpose()

```
template<typename T >
Matrix< T > Matrix< T >::Transpose ( ) const  [inline]
```

ez a fügvény a mátrix trasznponáltjával tér vissza

Returns

mátrix transzponáltja

3.2.3 Friends And Related Symbol Documentation

3.2.3.1 operator<<

```
template<typename T >
std::ostream & operator<< (
            std::ostream & os,
            const Matrix< T > & m )  [friend]
```

kiiró fugvény

Parameters

os	
m	

Returns

std::ostream&

The documentation for this class was generated from the following file:

- matrix.hpp

3.3 Network Class Reference

Public Member Functions

- **Network** (int net[ ], int len)
- void train (Matrix< double > Dataset[ ], Matrix< double > y[ ], int epoch, double lr)

  ez a fügvény szerepel az egész hálozat tanításáért
- Matrix< double > forward (const Matrix< double > &input)

  kiszámtija a neurális hálozat eredményét
- void save (const char *name) const

  menti az adott neurális hálózatot
- void load (const char *name)

  külső fájlból tölti be a neurálsi hálózat adatait

3.3.1 Member Function Documentation

3.3.1.1 forward()

```
Matrix< double > Network::forward (
            const Matrix< double > & x )
```

kiszámtija a neurális hálozat eredményét

Parameters

x	kiindulási adat

Returns

Matrix<double>

3.3.1.2 load()

```
void Network::load (
```

```
            const char * name )
```

külső fájlból tölti be a neurálsi hálózat adatait

Parameters

name	fájl név

3.3.1.3 save()

```
void Network::save (
            const char * name ) const
```

menti az adott neurális hálózatot

Parameters

name	fálj név

3.3.1.4 train()

```
void Network::train (
            Matrix< double > Dataset[],
            Matrix< double > y[],
            int epoch,
            double lr )
```

ez a fügvény szerepel az egész hálozat tanításáért

Parameters

Dataset	tanulási adat
y	kivánt kimenet
epoch	tanulás hossza
lr	tanulási ráta

The documentation for this class was generated from the following files:

- network.h
- network.cpp

Chapter 4

File Documentation

4.1 image.h

```
00001 #include "matrix.hpp"
00002
00003 Matrix<double> load_image(const char *fname);
```

4.2 layer.h

```
00001 #include "matrix.hpp"
00002 #ifndef LAYER_H
00003 #define LAYER_H
00004 class Layer{
00005     Matrix<double> weights, bias;
00006 public:
00007     Layer(int inputs=1, int outputs=1);
00008     //getterek
00009     Matrix<double> getW()const{return weights;}
00010     Matrix<double> getB()const{return bias;}
00011
00012     void setW(Matrix<double> w){weights = w;}
00013     void setB(Matrix<double> b){bias = b;}
00014
00015     Matrix<double> forward(const Matrix<double>& input);
00016     Matrix<double> backprop(const Matrix<double>& error, const Matrix<double>& prev_out,const
     Matrix<double>& z, double lr);
00017 };
00018 Matrix<double> sigmoid(const Matrix<double>& z);
00019 Matrix<double> sigmoid_prime(const Matrix<double>& z);
00020 #endif
```

4.3 main.cpp File Reference

neurális hálozat

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <chrono>
#include <iomanip>
#include <fstream>
#include "memtrace.h"
#include "matrix.hpp"
#include "layer.h"
#include "network.h"
#include "mnist.h"
#include "image.h"
```

Include dependency graph for main.cpp:

4.4 matrix.hpp

```
00001 #include <iomanip>
00002 #include <cstdlib>
00003 #include <ctime>
00004 #include <iostream>
00005 #include <random>
00006
00007 #ifndef MATRIX_H
00008 #define MATRIX_H
00009 template< typename T>
00010 class Matrix{
00011     int row, col;
00012     T **matrix;
00013 public:
00014     //kostruktorok
00021     Matrix(int row =1, int col=1):row(row),col(col){
00022         matrix = new T*[row];
00023         for(int i = 0; i< row; i++){
00024             matrix[i] = new T[col];
00025         }
00026     }
00032     Matrix(const Matrix<T> &rhs):row(rhs.row),col(rhs.col){
00033         matrix = new T*[row];
00034         for(int i = 0; i< row; i++){
00035             matrix[i] = new T[col];
00036             for(int j = 0; j < col; j++){
00037                 matrix[i][j] = rhs.matrix[i][j];
00038             }
00039         }
00040     }
00041     //destruktor
00046     ~Matrix(){
00047         for(int i = 0; i< row; i++){
00048             delete[] matrix[i];
00049         }
00050         delete[] matrix;
00051     }
00052     //getter
00058     int getRow()const{return row;}
00064     int getCol()const{return col;}
00065     //operators
00073     T& operator()(int r, int c){
00074         if(c >= col ||r >= row)
00075             throw std::out_of_range("out of range");
00076         return matrix[r][c];
00077     }
00078     const T operator()(int r, int c)const{
00079         if(c >= col ||r >= row)
00080             throw std::out_of_range("out of range");
00081         return matrix[r][c];
00082     }
00089     Matrix<T> operator+(const Matrix<T>& rhs)const{
00090         if(row != rhs.row || col != rhs.col){
00091             std::length_error("the shape doesn't match");
00092             std::cout«"("«row«","«col«") , ("«rhs.row«","«rhs.col«")\n";
00093         }
00094         Matrix<T> res(row, col);
00095         for(int i = 0; i<row; i++){
00096             for(int j = 0; j<col; j++){
00097                 res(i,j) = matrix[i][j] +rhs(i,j);
00098             }
00099         }
00100         return res;
00101     }
00108     Matrix<T> operator*(const Matrix<T>& rhs)const{
00109         if(row != rhs.row || col != rhs.col){
00110             std::length_error("the shape doesn't match");
00111             std::cout«"("«row«","«col«") , ("«rhs.row«","«rhs.col«")\n";
00112         }
00113         Matrix<T> res(row, col);
00114         for(int i = 0; i<row; i++){
00115             for(int j = 0; j<col; j++){
00116                 res(i,j) = matrix[i][j] * rhs(i,j);
00117             }
00118         }
00119         return res;
00120     }
00121     Matrix<T>& operator=(const Matrix<T>& rhs) {
00122         if (this == &rhs)
00123             return *this;
00124
00125         for (int i = 0; i < row; i++) {
00126             delete[] matrix[i];
00127         }
00128         delete[] matrix;
00129
```

```
00130         row = rhs.row;
00131         col = rhs.col;
00132         matrix = new T*[row];
00133         for (int i = 0; i < row; i++) {
00134             matrix[i] = new T[col];
00135             for (int j = 0; j < col; j++) {
00136                 matrix[i][j] = rhs(i, j);
00137             }
00138         }
00139         return *this;
00140     }
00146     Matrix<T> operator-(const Matrix<T>& rhs){
00147         if(row != rhs.row || col != rhs.col)
00148             std::cout«"-the shape doesn't match"«"("«row«","«col«") , ("«rhs.row«","«rhs.col«")\n";
00149 
00150         Matrix<T> res(row, col);
00151         for(int i = 0; i<row; i++){
00152             for(int j = 0; j<col; j++){
00153                 res(i,j) = matrix[i][j] - rhs(i,j);
00154             }
00155         }
00156         return res;
00157     }
00164     friend std::ostream& operator«(std::ostream &os, const Matrix<T>& m){
00165         for(int i = 0; i< m.row; i++){
00166             os«"[ ";
00167             for(int j = 0; j < m.col; j++){
00168                 os«std::setprecision(3)«m.matrix[i][j]«" ";
00169             }
00170             os«"]\n";
00171         }
00172         return os;
00173     }
00174     //tagfugvenyek
00179     void mrand(){
00180         std::random_device rd;
00181         std::mt19937 gen(rd());
00182         std::uniform_real_distribution<double> dis(-0.5, 0.5);
00183 
00184         for(int i = 0; i < row; i++) {
00185             for(int j = 0; j < col; j++) {
00186                 matrix[i][j] = dis(gen);
00187             }
00188         }
00189     }
00194     void zeros(){
00195         for(int i = 0; i<row; i++){
00196             for(int j = 0; j<col; j++){
00197                 matrix[i][j] = 0;
00198             }
00199         }
00200     }
00206     Matrix<T> Transpose()const{
00207         Matrix<T> res(col, row);
00208         for(int i = 0; i<row; i++){
00209             for(int j = 0; j<col; j++){
00210                 res(j,i) = matrix[i][j];
00211             }
00212         }
00213         return res;
00214     }
00215 };
00224 template<typename T>
00225 Matrix<T> dot(const Matrix<T>& lhs, const Matrix<T>& rhs) {
00226     if (lhs.getCol() != rhs.getRow()) {
00227         std::cout « "Dot product: the shapes don't match (" « lhs.getRow() « "," « lhs.getCol() « ") ,
      (" « rhs.getRow() « "," « rhs.getCol() « ")\n";
00228         throw std::invalid_argument("The shapes don't match");
00229     }
00230 
00231     Matrix<T> res(lhs.getRow(), rhs.getCol());
00232 
00233     for (int i = 0; i < lhs.getRow(); i++) {
00234         for (int j = 0; j < rhs.getCol(); j++) {
00235             T temp = 0.0;
00236             for (int k = 0; k < lhs.getCol(); k++) {
00237                 temp += lhs(i, k) * rhs(k, j);
00238             }
00239             res(i, j) = temp;
00240         }
00241     }
00242     return res;
00243 }
00244 
00253 template<typename T>
00254 Matrix<T> operator-(T lhs, const Matrix<T>& rhs){
00255     Matrix<T> res(rhs.getRow(),rhs.getCol());
```

```
00256     for(int i = 0; i<rhs.getRow(); i++){
00257         for(int j = 0; j<rhs.getCol(); j++){
00258             res(i,j) = lhs-rhs(i,j);
00259         }
00260     }
00261     return res;
00262 }
00271 template<typename T>
00272 Matrix<T> operator*(T lhs, const Matrix<T>& rhs){
00273     Matrix<T> res(rhs.getRow(),rhs.getCol());
00274     for(int i = 0; i<rhs.getRow(); i++){
00275         for(int j = 0; j<rhs.getCol(); j++){
00276             res(i,j) = lhs*rhs(i,j);
00277         }
00278     }
00279     return res;
00280 }
00288 template<typename T>
00289 T sum(const Matrix<T>& m){
00290     T Sum = 0.0;
00291     for(int i = 0; i<m.getRow(); i++){
00292         for(int j = 0; j<m.getCol(); j++){
00293             Sum += m(i,j);
00294         }
00295     }
00296     return Sum;
00297 }
00298 #endif
```

4.5 mnist.h

```
00001 #include "matrix.hpp"
00002
00003 void MNIST_images(Matrix<double> x[], int len);
00004 void MNIST_labels(Matrix<double> y[], int len);
```

4.6 network.h

```
00001 #include "matrix.hpp"
00002 #include "layer.h"
00003
00004 class Network{
00005     Layer *layers;
00006     int len;
00007 public:
00008     Network(int net[], int len);
00009     Network();
00010     void train(Matrix<double> Dataset[],Matrix<double> y[], int epoch, double lr);
00011     Matrix<double> forward(const Matrix<double>& input);
00012     void save(const char* name)const;
00013     void load(const char* name);
00014     ~Network();
00015 };
```

Index