”

"

67315
4C++

22:00

Introduction to C++,Classes,Operator Overloading,References,Rule of Three,Exceptions

Gmail(Amazon Alexa, Apple Siri)(Artificial neural network)

32.21.2

https://www.youtube.com/watch?v=aircAruvnKk

Fully Connected

- 1.2.2
-
-
- 2.2.2
- 2.2.3

$y \in \mathbb{R}^n x \in \mathbb{R}^m$

$$y = f\left(W \cdot x + b\right)$$

- **Weights**$W \in M_{n \times m}(\mathbb{R})$
- **Bias**$b \in \mathbb{R}^n$
- 1.2.3$f : \mathbb{R}^n \to \mathbb{R}^n$
- $y = f(W \cdot x + b) \in \mathbb{R}^n x \in \mathbb{R}^m$
  $f : \mathbb{R}^n \to \mathbb{R}^n$
- **ReLU**

$$\forall x \in \mathbb{R} \qquad ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & else \end{cases}$$

$x \in \mathbb{R}^n$

- **Softmax**

$$\forall x \in \mathbb{R}^n \qquad Softmax(x) = \frac{1}{\sum_{k=1}^{n} e^{x_k}} \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ e^{x_3} \\ \vdots \\ e^{x_n} \end{bmatrix} \in \mathbb{R}^n$$

$x \in \mathbb{R}^n k x_k$
.cmathstd::exp$exp(t) e^t$

$x \in \mathbb{R}^n$

float

float (32-bit)

float

$$(a+b)+c = a+(b+c)$$

$$(A \cdot B)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

•

	Bias	**Weights**	
$Relu$	$b_1 \in \mathbb{R}^{128}$	$W_1 \in M_{128\times784}$	
$Relu$	$b_2 \in \mathbb{R}^{64}$	$W_2 \in M_{64\times128}$	
$Relu$	$b_3 \in \mathbb{R}^{20}$	$W_3 \in M_{20\times64}$	
$Softmax$	$b_4 \in \mathbb{R}^{10}$	$W_4 \in M_{10\times20}$	

•

$$\begin{aligned} r_1 &= Relu(W_1 \cdot x + b_1) \\ r_2 &= Relu(W_2 \cdot r_1 + b_2) \\ r_3 &= Relu(W_3 \cdot r_2 + b_3) \\ r_4 &= Softmax(W_4 \cdot r_3 + b_4) \end{aligned}$$

- $2.2.2x$
- $i+1ir_i$
- $2.2.3r_4$

- $A \in M_{28\times28}\,([0,1])$(Grayscale)$28 \times 28A$
- plot_img.py
- $28 \cdot 28 = 748$

- 101
-
-
-

-
-

Value		0.003	0.08					0.9	0.007	0.01
Index										

90%7

main.cpp

$./mlpnetwork w1 w2 w3 w4 b1 b2 b3 b4

- iw_i
- ib_i

```
ex4_cpp/Ex4_CPP_2023-master/lyx_stuff
```

-
- privatepublicAPI
-
-
- (standalone, non-member function)(member function)by reference by value const
- **.std::vector C++STLSTL**

float n**Matrix**

Description	**Name**	**Comments**
	Constructors	
Constructor	Matrix(int rows, int cols)	Constructs Matrix of size **rows**×**cols**. Inits all elements to **0**.
Default Constructor	Matrix()	Constructs Matrix of size 1×1. Inits the single element to **0**.
Copy Constructor	Matrix(Matrix m)	Constructs matrix from another Matrix m.
Destructor	~Matrix()	

Methods & Functions		
Getter	get_rows()	returns the amount of rows as int.
Getter	get_cols()	returns the amount of cols as int.

	transpose()	Transforms a matrix into its **transpose** matrix,
		i.e $(A.transpose())_{ij} = A_{ji}$.
		Supports function call chaining/concatenation (), for example:
		Matrix a(5,4), b(4,5);
		a.transpose();// a.get_rows == 4, a.get_cols == 5
		b.transpose().transpose();// b is same as before after calling transpose twice (chaining)

	vectorize()	Transforms a matrix into a column vector(section 3.1.2).
		Supports function call chaining/concatenation. For example:
		Matrix m(5,4); ...
		m.vectorize();
		m.get_cols() == 1
		m.get_rows() == 20
		m.transpose().vectorize().transpose(); // an example of function call chaining/concatenation

	plain_print()	Prints matrix elements, no return value. Prints space after each element (including last element in row). Prints newline after each row (including last row).
	dot(Matrix m)	Returns a matrix which is the **elementwise multiplication** (Hadamard product) of this matrix with another matrix m: $\forall i, j : (A.dot(B))_{ij} = A_{ij} \cdot B_{ij}$
	norm()	Returns the Frobenius norm of the given matrix:$A.norm() = \sqrt{\sum_{i,j} A_{ij}^2}$

	rref()	Returns a new Matrix that is the reduced row echelon form of the original. The original Matrix should be unchanged (see section 3.1.4)

Operators		
+	Matrix addition	Matrix a, b;→ a + b
=	Assignment	Matrix a, b;→ a = b
*	Matrix multiplication	Matrix a, b;→ a * b
*	Scalar multiplication on the right	Matrix m; float c;→ m * c
*	Scalar multiplication on the left	Matrix m; float c;→ c * m
+=	Matrix addition accumulation	Matrix a, b;→ a += b

()	Parenthesis indexing	For cell (i,j) in matrix m:
		m(i,j) will return the element in cell (i,j) in matrix m.
		For example:
		Matrix m(5,4);
		m(1,3) = 10;
		float x = m(1,3); // x equals 10.0

[]	Brackets indexing	Let m be a matrix with i rows and j columns. Then m has i*j cells.
		For every 0<=k<i*j
		m[k] will return the k'th element as if m is represented as a vector (see section 3.1.2).
		For example:
		Matrix m(5,4);
		m[3] = 10;
		float x = m[3]; // x equals 10.0

<<	Output stream	Pretty print of matrix as (see section 3.1.1)
>>	Input stream	Fills matrix elements: has to read input streleadam fully, otherwise it's an error (don't trust the user to validate it) (see section 3.1.3)

$<<A$

```
for i = 1 to A.rows:
        for j = 1 to A.cols: lead
                if A(i,j) > 0.1:
                        print"**" (double asterisk)
                else:
                        print"  " (double space)
        print newline
```

A

$$A(i,j) == A[i \cdot \text{rowsize} + j]$$

- i
- j
- $rowsize$

$A \in M_{3\times 4}$

$$A(2,1) == A[2 \cdot 4 + 1] == A[9]$$

\char*std::istream::readInput stream

https://www.tutorialspoint.com/reading-and-writing-binary-file-in-c-cplusplus

https://www.cplusplus.com/reference/istream/istream/read/

char*casting

Reduced Row Echelon Form

Instructions:

-This method returns a matrix that is the reduced row echelon form of the original matrix. Note that the original matrix must remain unchanged.

-To find the reduced row echelon form of a matrix, we do Gaussian elimination to get a matrix with the following properties:

1. For each row (that is non-zero), the leading non-zero digit is 1.

2. For each leading digit 1, all cells of the matrix in the same column are zeroes.

3. The leading digit 1 of every non-zero row is to the right of the leading digit 1 of every row above it.

4. All rows of only zeroes are found at the bottom of the matrix.

-**For example:**

$$\begin{bmatrix} 2 & 4 & 8 & 1 \\ 16 & 3 & 5 & 0 \\ 1 & 0 & 4 & 3 \end{bmatrix} \xrightarrow{\text{RREF}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1.25 \\ 0 & 0 & 1 & 0.75 \end{bmatrix}$$

(Original Matrix) (Reduced Row Echelon Form matrix)

Tip:

-split up each of the Gaussian Elimination operations (row switch, multiplication by scalar, row subtraction, etc.) each into their own functions. This will make it much easier to debug.

Notes:

1) The matrices returned may have "negative zeroes" instead of regular zeroes (-0 instead of 0) in some entries. In C/C++, zero and negative zero are equivalent, so you can ignore this. This has to do with the way floating point numbers are represented under the hood. For further reading see the link given in the list of links below.
2) These functions should be implemented in one of the .cpp files in your project, not in the header files.

-For further reading and testing see the following links:

RREF Matrix calculator:

https://www.emathhelp.net/calculators/linear-algebra/reduced-row-echelon-form-rref-calculator/?i=

RREF Wikipedia Reference:

https://en.wikipedia.org/wiki/Row_echelon_formReduced_row_echelon_form

Signed Zero Wikipedia Reference:

https://en.wikipedia.org/wiki/Signed_zero

activation::softmaxactivation::relunamespace activationsoftmaxreluActivation**Activation**
typedef function pointers
Dense

Description	**Name**	**Comments**
Constructor	Dense(weights, bias, ActivationFunction)	Inits a new layer with given parameters. C'tor accepts 2 matrices and activation function
	Methods	
Getter	get_weights()	Returns the weights of this layer.
Getter	get_bias()	Returns the bias of this layer.
Getter	get_activation()	Returns the activation function of this layer.
	Operators	
()	Parenthesis	Applies the layer on input and returns output matrix Layers operate as per section 2.2.1 e.g: Dense layer(w, b, act); Matrix output = layer(input);

2.2.1**MlpNetwork**
struct digit

Description	**Name**	**Comments**
Constructor	MlpNetwork(weights[], biases[])	Accepts 2 arrays of matrices, size 4 each. One for weights and one for biases. Constructs the network described (sec. 2.2).
	Operators	
()	Parenthesis	Applies the entire network on input. Returns the digit struct. MlpNetwork m(...); digit output = m(img);

exceptions

•

•

exception –

std::length_error ∗

std::out_of_range ∗

std::runtime_error ∗

std::bad_alloc ∗

bad_alloc**https://en.cppreference.com/w/cpp/memory/new/bad_alloc**

–

Makefile

```
make mlpnetwork
```

.2.3
presubmit**Presubmit**

~proglab/presubmit/ex4/srcs/presubmit.cpp

6

Matrix.h	Matrix.cpp
Activation.h	Activation.cpp
Dense.h	Dense.cpp
MlpNetwork.h	MlpNetwork.cpp

-
- math.h<cmath>`char*std::stringfreemallocdeletenew`CC++**C++**
- **std::vectorC++STLSTL**
- valgrind
- by referenceby value**reference**
- const**const**
- Pre-submission Script**Pre-submission**
-
-
-
-