TensorLab Wiki

这是一个基于 C++和 Microsoft GDI+ API 的人工神经网络 demo 程序。该文档包含了这个程序的数据结构，函数，命令以及类的使用说明。

This is artificial neural network demo program based on C++ and Microsoft GDI+ API. This document includes the introduction of this program’s data structures, functions, commands, and classes.

数据结构 Data Structure

这个小节将以数据结构为主要对象进行说明，包括了这些结构的所在头文件、常量别名、字段以及函数成员。构造函数省略介绍了移动构造、复制构造函数以及隐式函数，末尾给出了派生。在自定义派生时，可以参考结构字段以及成员函数进行。这些结构一些是 C 语言结构。

In this chapter, data structure introduction will be the main topic including the head file these structures included, constant alias, fields, and function members. Move, copy constructor and implicit functions have been omitted in introduction. Derivation is at last of each structure introduction as a reference of derivation customization. Some of these structures are in C language.

bmio

bmio::px

matrix

mtx::mtx_pos

mtx::mtx_info

mtx::mtx_extm

netbatlib

BN

fc::FCBN

conv::ConvBN

ada::AdaDeltaVect

ada::AdaDeltaVal

layer

layer::Layer

layer::LayerFC

layer::LayerFCBN

layer::LayerConv

layer::LayerConvBN

layer::LayerPool

layer::LayerTrans

bmio

头文件，位图库.

Head file, bitmap library.

bmio::px

像素

Pixel

常量 Constant

BMIO_PX

字段 Field

u_char r

用于存储 RGBA 红色数值

For RGBA red value storage

u_char g

用于存储 RGBA 绿色数值

For RGBA green value storage

u_char b

用于存储 RGBA 蓝色数值

For RGBA blue value storage

u_char a

用于存储 RGBA Alpha 通道数值

For RGBA Alpha channel value storage

matrix

头文件，矩阵库.

Head file, matrix library.

mtx::mtx_pos

矩阵元素位置

Matrix element position

常量 Constant

MATRIX_POS

字段 Field

uint64_t ln

行

Line

uint64_t col

列

Column

mtx::mtx_info

矩阵信息

Matrix information

常量 Constant

MATRIX_INFO

字段 Field

MATRIX mtx_val

矩阵指针

Matrix pointer

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

mtx::mtx_extm

矩阵元素极值

Matrix element extremum

常量 Constant

MATRIX_EXTREME

字段 Field

double val

极值

Extremum

net_list<mtx_pos> pos_list

极值元素所在位置列表

Elements’ extremum position list

netbatlib

头文件，网络批量处理库.

Head file, network batch-process library.

BN

批归一化输出

Batch-normalization output

常量 Constant

BN_PTR = std::shared<BN>

派生 Derivation

fc::FCBN

conv::ConvBN

fc::FCBN

全连接批归一化输出 Batch-normalization output for fully connection

常量 Constant

BN_CONV

字段 Field

vect vecMiuBeta

期望

Expectation

vect vecSigmaSqr

方差

Variant

set<vect> setBarX

归一化值

Normalization value

set<vect> setY

归一化输出

Normalization output

conv::ConvBN

卷积批归一化输出

Batch-normalization output for convolution

常量 Constant

BN_CONV

字段 Field

feature vecMiuBeta

期望

Expectation

feature vecSigmaSqr

方差

Variant

set<feature> setBarX

归一化值

Normalization value

set<feature> setY

归一化输出

Normalization output

ada::AdaDeltaVect

AdaDelta 向量数据结构，用于辅助梯度对权重等进行更新

AdaDelta vector data structure, use to optimize gradient updating weight, etc.

字段 Field

double dRho

衰减控制

Decay controller

double dEpsilon

除零占位

Zero devisor dominant

vect vecPreDelta

前次更新差值

Previous updating difference

函数 Function

AdaDeltaVect

构造函数

Constructor

参数 Parameter

uint64_t nSizeLnCnt

梯度行计数

Line count of gradient

uint64_t nSizeColCnt

梯度列计数

Column count of gradient

Delta

获取更新用差值

Get difference for updating

参数 Parameter

vect vecCurrGrad

当前梯度向量，一般为损失到需更新变量的梯度

Current gradient vector, the gradient from loss to the variable needed update commonly

返回 Return

用于更新目标值的差值

Difference for target value needed to update

Reset

重置数据

Reset the data

示例 Example

C++

```
// 为全连接层具有 4 个神经元的权重声明一个 AdaDelta 数据结构，衰减与除零占位默认
// Declare and instance an AdaDelta data structure for a fully connection layer weight
with 4 neurons.
```

```
ada::AdaDeltaVect adaFCWeight(4, 1);
// 前向传播并反向传播
// Forward and back propagation
/* code */
auto vecLayerOutput = fc::Output(vecInput, vecWeight);
auto vecLayerActivate = sigmoid(vecLayerOutput);
…
// 获取到损失到权重的梯度
// Get gradient from loss to weight
auto vecGradLossToOutput = vecGradLossToActivate.elem_cal_opt(sigmoid_derivative(
                                                                vecLayerOutput));
auto vecGradLossToWeight = fc::GradLossToWeight(vecGradLossToOutput, vecInput);
// 获取梯度的 AdaDelta 更新值并更新权重
// Get AdaDelta updating value of gradient and update the weight
vecWeight -= adaFCWeight.Delta(vecGradLossToWeight);
// 其他需要更新的目标值均可使用 AdaDelta 进行更新
// Other all values needed to update could use AdaDelta to update in this way
```

ada::AdaDeltaVal

AdaDelta 数值数据结构，用于辅助梯度对权重等进行更新

AdaDelta value data structure, use to optimize gradient updating weight, etc.

字段 Field

double dRho

衰减控制

Decay controller

double dEpsilon

除零占位

Zero devisor dominant

double dPreDelta

前次更新差值

Previous updating difference

函数 Function

AdaDeltaVal

构造函数

Constructor

参数 Parameter

double dRhoVal

初始化衰退控制

Decay controller initializer

double dEpsilonVal

除零占位初始化

Zero divisor dominant initializer

Delta

获取更新用差值

Get difference for updating

参数 Parameter

double dCurrGrad

当前梯度数值，一般为损失到需更新变量的梯度

Current gradient value, the gradient from loss to the variable needed update commonly

返回 Return

用于更新目标值的差值

Difference for target value needed to update

layer

头文件，网络层.

Head file, network layer.

layer::Layer

神经网络层

Neural network layer

常量 Constant

LAYER_PTR 智能指针变量 Smart pointer variable

字段 Field

uint64_t iActFuncType

激活函数类型

Activation function type

uint64_t iLayerType

层类型

Layer Type

double dLayerLearnRate

层学习率

Layer learning rate

函数 Function

Layer

构造函数

Constructor

参数 Parameter

uint64_t iLayerTypeVal

层类型变量

Layer type variable

[FC]　　全连接

[CONV]　　卷积

[POOL]　　池化

[FC_BN]　　全连接批归一化

[CONV_BN]　　卷积批归一化

[TRANS] 向量变换

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

double dLearnRate

学习率

Learning Rate

Activate

激活操作

Activation operation

参数 Parameter

set<T> setInput

输入集合

Input set

返回 Return

激活结果值

Activation result value

Derivative

导数反向计算

Back derivative calculation

参数 Parameter

set<T> setActInput

激活输入集合，当激活函数是 softmax 时为激活输出集合

Activation input set, change to activation output set when softmax is the activation function

set<T> setGrad

损失到激活输出的梯度集合，当激活函数是 softmax 时为标签分类向量集合

Gradient set from loss to activation output, change to label classification vector set when softmax is the activation function

返回 Return

损失到激活输入的梯度集合

Gradient set from loss to activation input

派生 Derivation

layer::LayerFC

layer::LayerFCBN

layer::LayerConv

layer::LayerConvBN

layer::LayerPool

layer::LayerTrans

layer::LayerFC

全连接层

Layer for fully connection

常量 Constant

LAYER_FC

字段 Field

vectvecLayerWeight

层权重

Layer weight

set<vect> setLayerInput

层输入

Layer input

set<vect> setLayerOutput

层输出

Layer output

_ADA AdaDeltaVect advLayerDelta

AdaDelta 权重数据

AdaDelta data for weight

函数 Function

LayerFC

构造函数

Constructor

参数 Parameter

uint64_t iInputLnCnt

输入向量行计数

Line count of input vector

uint64_t iOutputLnCnt

输入向量列计数

Column count of input vector

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

double dLearnRate

学习率

Learning Rate

double dRandBoundryFirst

初始化伪随机数范围区间第一个区间值

First interval value of pseudo random number for initialization

double dRandBoundrySecond

初始化伪随机数范围区间第二个区间值

Second interval value of pseudo random number for initialization

double dAcc

伪随机数精度

Pseudo random number accuracy

ForwProp

当前层前向传播

Forward propagation for current layer

参数 Parameter

set<vect> setInput

输入向量集合

Input vector set

bool bFirstLayer

第一层标识

First layer signal

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

当前层前反向传播

Forward propagation for current layer

参数 Parameter

set<vect> setGradOrOutput

损失到当前层激活输出的梯度，激活函数为 softmax 时为当前层激活输出向量集合

Vector set of gradients from loss to current layer’s activation output, change to current layer’s activation output when softmax is activation function

set<vect> setOrigin

分类标签值向量集合

Vector set of classification label value

返回 Return

反向传播输出损失到上一层激活输出的梯度

Gradient from loss to last layer’s activation output of back propagation output

Reset

重置当前层数据

Reset current layer data

layer::LayerFCBN

全连接批归一化层

Batch-normalization layer for fully connection

常量 Constant

LAYER_FC_BN

字段 Field

double dBeta

偏移度

Shift

double dGamma

尺度

Scale

double dEpsilon

除零占位

Zero divisor dominant

set<vect> setLayerInput

层输入向量集合

Layer input vector set

_ADA AdaDeltaVal advBeta

偏移度 AdaDelta 数据

AdaDelta data for shift

_ADA AdaDeltaVal advGamma

尺度 AdaDelta 数据

AdaDelta data for scale

BN_FC BNData

全连接批归一化输出数据

Batch-normalization output data for fully connection

函数 Function

LayerFCBN

构造函数

Constructor

参数 Parameter

uint64_t dShift

偏移度初始化数值

Value for shift initialization

uint64_t dScale

尺度初始化数值

Value for scale initialization

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

double dLearnRate

学习率

Learning Rate

double dDmt

除零占位数值

Zero divisor dominant value

ForwProp

当前层前向传播

Forward propagation for current layer

参数 Parameter

set<vect> setInput

输入向量集合

Input vector set

bool bFirstLayer

第一层标识

First layer signal

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

当前层前反向传播

Forward propagation for current layer

参数 Parameter

set<vect> setGradOrOutput

损失到当前层激活输出的梯度，激活函数为 softmax 时为当前层激活输出向量集合

Vector set of gradients from loss to current layer’s activation output, change to current layer’s activation output when softmax is activation function

set<vect> setOrigin

分类标签值向量集合

Vector set of classification label value

返回 Return

反向传播输出损失到上一层激活输出的梯度

Gradient from loss to last layer’s activation output of back propagation output

Reset

重置当前层数据

Reset current layer data

layer::LayerConv

卷积层

Layer for convolution

常量 Constant

LAYER_CONV

字段 Field

uint64_t iLayerLnStride

层行向步幅

Layer line directional stride

uint64_t iLayerColStride

层列向步幅

Layer column directional stride

uint64_t iLayerLnDilation

层行向扩张

Layer line directional dilation

uint64_t iLayerColDilation

层列向扩张

Layer column directional dilation

uint64_t iLayerInputPadTop

层输入向量顶部扩展

Padding of layer input vector top side

uint64_t iLayerInputPadRight

层输入向量右部扩展

Padding of layer input vector right side

uint64_t iLayerInputPadBottom

层输入向量底部扩展

Padding of layer input vector bottom side

uint64_t iLayerInputPadLeft

层输入向量左部扩展

Padding of layer input vector left side

uint64_t iLayerLnDistance

层行方向元素间距

Layer line directional distance of each element

uint64_t iLayerColDistance

层列方向元素间距

Layer column directional distance of each element

tensor tenKernel

卷积核张量集合

Convolution kernel tensor set

set<feature> setLayerInput

层输入张量集合

Tensor set of layer input

set<feature> setLayerOutput

层输出张量集合

Tensor set of layer output

_ADA ada_tensor<_ADA AdaDeltaVect> advLayerDelta

层卷积核 AdaDelta 数据张量集合

AdaDelta data tensor set of layer kernels

函数 Function

LayerConv

构造函数

Constructor

参数 Parameter

uint64_t iKernelAmt

卷积核个数

Kernel amount

uint64_t iKernelChannCnt

卷积核通道计数

Kernel channel count

uint64_t iKernelLnCnt

卷积核行计数

Kernel line count

uint64_t iKernelColCnt

卷积核列计数

Kernel column count

uint64_t iLnStride

行向步幅

Line directional stride

uint64_t iColStride

列向步幅

Column directional stride

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

double dLearnRate

学习率

Learning Rate

double dRandBoundryFirst

初始化伪随机数范围区间第一个区间值

First interval value of pseudo random number for initialization

double dRandBoundrySecond

初始化伪随机数范围区间第一个区间值

Second interval value of pseudo random number for initialization

double dRandBoundryAcc

伪随机数精度

Pseudo random number accuracy

uint64_t iLnDilation

行向扩张

Line directional dilation

uint64_t iColDilation

列向扩张

Column directional dilation

uint64_t iInputPadTop

输入向量顶部扩展

Padding for input vector top side

uint64_t iInputPadRight

输入向量右部扩展

Padding for input vector right side

uint64_t iInputPadBottom

输入向量底部扩展

Padding for input vector bottom side

uint64_t iInputPadLeft

输入向量左部扩展

Padding for input vector left side

uint64_t iLnDistance

行方向元素间距

Line directional distance for each element

uint64_t iColDistance

列方向元素间距

Column directional distance for each element

ForwProp

当前层前向传播

Forward propagation for current layer

参数 Parameter

set<feature> setInput

输入张量集合

Input tensor set

bool bFirstLayer

第一层标识

First layer signal

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

当前层前反向传播

Forward propagation for current layer

参数 Parameter

set<feature> setGrad

损失到当前层激活输出的梯度张量集合

Tensor set of gradients from loss to current layer’s activation output

返回 Return

反向传播输出损失到上一层激活输出的梯度

Gradient from loss to last layer’s activation output of back propagation output

Reset

重置当前层数据

Reset current layer data

layer::LayerConvBN

卷积批归一化层

Batch-normalization layer for convolution

常量 Constant

LAYER_CONV_BN

字段 Field

vect vecBeta

偏移

Shift

vect vecGamma

尺度

Scale

double dEpsilon

除零占位

Zero divisor dominant

set<feature > setLayerInput

层输入张量集合

Layer input tensor set

_ADA AdaDeltaVect advBeta

偏移度 AdaDelta 数据

AdaDelta data for shift

_ADA AdaDeltaVect advGamma

尺度 AdaDelta 数据

AdaDelta data for scale

BN_CONV BNData

卷积批归一化输出数据

Batch-normalization output data for convolution

函数 Function

LayerConvBN

构造函数

Constructor

参数 Parameter

uint64_t iChannCnt

通道计数

Channel count

uint64_t dShift

偏移度初始化数值

Value for shift initialization

uint64_t dScale

尺度初始化数值

Value for scale initialization

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

double dLearnRate

学习率

Learning Rate

double dDmt

除零占位数值

Zero divisor dominant value

ForwProp

当前层前向传播

Forward propagation for current layer

参数 Parameter

set<feature> setInput

输入张量集合

Input tensor set

bool bFirstLayer

第一层标识

First layer signal

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

当前层前反向传播

Forward propagation for current layer

参数 Parameter

set<feature> setGrad

损失到当前层激活输出的梯度张量集合

Tensor set of gradients from loss to current layer’s activation output

返回 Return

反向传播输出损失到上一层激活输出的梯度

Gradient from loss to last layer’s activation output of back propagation output

Reset

重置当前层数据

Reset current layer data

layer::LayerPool

池化层

Layer for pooling

常量 Constant

LAYER_POOL

字段 Field

uint64_t iPoolType

池化类型

Pooling type

uint64_t iLayerFilterLnCnt

层过滤行向计数

Layer filter line directional count

uint64_t iLayerFilterColCnt

层过滤列向计数

Layer filter column directional count

uint64_t iLayerLnStride

层行向步幅

Layer line directional stride

uint64_t iLayerColStride

层列向步幅

Layer column directional stride

uint64_t iLayerLnDilation

层行向扩张

Layer line directional dilation

uint64_t iLayerColDilation

层列向扩张

Layer column directional dilation

set<feature> setLayerInput

层输入张量集合

Tensor set of layer input

set<feature> setLayerOutput

层输出张量集合

Tensor set of layer output

函数 Function

PoolUpType

获取上池化类型

Get up pooling type

参数 Parameter

uint64_t iPoolIdx

池化索引类型

Pooling type index

[POOL_AVG] 平均池化 Average pooling

[POOL_MAX] 最大池化 Max pooling

[POOL_GAG] 全局平均池化 Global average pooling

返回 Return

[POOL_UP_AVG] 平均上池化 Average up pooling

[POOL_ UP_MAX] 最大上池化 Max up pooling

[POOL_ UP_GAG] 全局平均上池化 Global average up pooling

LayerPool

构造函数

Constructor

参数 Parameter

uint64_t iPoolTypeVal

池化类型

Pooling type

[POOL_AVG] 平均池化 Average pooling

[POOL_MAX] 最大池化 Max pooling

[POOL_GAG] 全局平均池化 Global average pooling

uint64_t iFilterLnCnt

过滤行计数

Filter line count

uint64_t iFilterColCnt

过滤列计数

Filter column count

uint64_t iLnStride

行向步幅

Line directional stride

uint64_t iColStride

列向步幅

Column directional stride

uint64_t iLnDilation

行向扩张

Line directional dilation

uint64_t iColDilation

列向扩张

Column directional dilation

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

ForwProp

当前层前向传播

Forward propagation for current layer

参数 Parameter

set<feature> setInput

输入张量集合

Input tensor set

bool bFirstLayer

第一层标识

First layer signal

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

当前层前反向传播

Forward propagation for current layer

参数 Parameter

set<feature> setGrad

损失到当前层激活输出的梯度张量集合

Tensor set of gradients from loss to current layer’s activation output

返回 Return

反向传播输出损失到上一层激活输出的梯度

Gradient from loss to last layer’s activation output of back propagation output

Reset

重置当前层数据

Reset current layer data

layer::LayerTrans

向量转换层

Layer for vector transformation

常量 Constant

LAYER_TRANS

字段 Field

uint64_t iLnCnt

矩阵行计数

Line count

uint64_t iColCnt

矩阵列计数

Column count

bool bFeatToVec

张量转向量标识

Feature to vector transformation signal

函数 Function

LayerTrans

参数 Parameter

层输入值为张量

Tensor is layer input

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

参数 Parameter

层输入值为向量

Vector is layer input

uint64_t iChannLnCnt

输入通道行计数

Input channel line count

uint64_t iChannColCnt

输入通道列计数

Input channel column count

uint64_t iActFuncTypeVal

激活函数类型变量

Activation function type value

[NULL][SIGMOID][RELU][SIGMOID]

ForwProp

参数 Parameter

层输入值为张量

Tensor is layer input

set<feature> setInput

输入张量集合

Input tensor set

返回 Return

前向传播激活输出

Activation output of forward propagation

参数 Parameter

层输入值为向量

Vector is layer input

set<vect> setInput

输入向量集合

Input vector set

返回 Return

前向传播激活输出

Activation output of forward propagation

BackProp

参数 Parameter

层输入值为张量

Tensor is layer input

set<vect> setGrad

损失到当前层输出的梯度向量集合

Vector set of gradients from loss to current layer’s output

返回 Return

前向传播激活输出

Activation output of forward propagation

参数 Parameter

层输入值为向量

Vector is layer input

set<feature> setGrad

损失到当前层输出的梯度张量集合

Tensor set of gradients from loss to current layer’s output

返回 Return

前向传播激活输出

Activation output of forward propagation

函数 Function

这个小节将以函数为主要对象进行说明，通过头文件的形式将函数进行排列。函数包括了基本数据分析处理，激活函数以及神经网络的基础函数。函数介绍包括参数，返回以及具体使用方法。一些较为抽象的函数会给出示例。

In this chapter, the function will be the main topic and arranged these functions by head file. These functions ranges from basic data analysis & process to activation function and neural network fundamental function. Function introduction includes parameters, return and details for usage. Some intricate function will provide example.

bagrt

bagrt::num_cnt
bagrt::quick_sort
bagrt::reset_ptr
bagrt::random_number
bagrt::input_paragraph
bagrt::extract_number
bagrt::charset_exchange
bagrt::primes
bagrt::primes_fact
bagrt::greatest_common_divisor
bagrt::least_common_multiple

csvio

csvio::input_table
csvio::output_table

netbaslib

fc::Output
fc::GradLossToInput
fc::GradLossToWeight
fc::FeatureTransform
conv::Conv
conv::GradLossToKernel
conv::GradLossToInput
conv::PoolDown
conv::PoolUp

netbatlib

fc::BNTrain
fc::BNGradLossToInput
fc::BNGradLossToScale
fc::BNGradLossToSshift
fc::BNDeduce
conv::BNTrain
conv::BNGradLossToInput
conv::BNGradLossToScale
conv::BNGradLossToSshift
conv::BNDeduce

funclib

sigmoid
sigmoid_dv
ReLU
ReLU_dv
softmax
softmax_dv
cec_grad
softmax_cec_grad

bagrt

头文件，基础算法库.

Head file, basic algorithm library.

bagrt::num_cnt

计算离散闭区间的元素个数

Calculate the element amount between a discrete closed interval

参数 Parameters

uint64_t from

开始位置

Begin index

uint64_t to

结束位置

End index

返回 Return

计算结果

Answer of calculation

bagrt::quick_sort

排序算法

Sort algorithm

参数 Parameters

std::unique_ptr seq_val

序列指针

Sequence pointer

uint64_t begin

开始位置

Begin index

uint64_t end

结束位置

End index

bool asc

升序信号

Ascent signal

std::function func_comp

升序比较定义函数

Definition function of ascent comparation

返回 Return

排序结果

Sort result

示例 Example

C++

```
// 已分配内存含有元素的乱序列指针
// Memory allocated irregular sequence pointer with elements
std::unique_ptr<T> seq(len);
/* code */
```

…

```
// 定义前者大于后者为升序
// Define the previous one bigger than the later as ascent
// 定义方式 #1
// Definition #1
template<typename fT, typename sT> bool compare_ascent(fT &first, st &second) { return first > second; }
// 升序
// Ascent
quick sort(seq, 0, len, true, compare_ascent);
// 定义方式 #2
// Definition #2
// 升序，使用 lambda 函数
// Ascent, use lambda function
quick sort(seq, 0, len, true, [](fT &first, st &second) { return first > second; });
```

bagrt::reset_ptr

重置指针

Reset the pointer

参数 Parameters

std::unique_ptr ptr …

多个序列指针

Multiple sequence pointer

示例 Example

C++

```
// 多个指针
// Several pointers
std::unique_ptr<T> a(len);
std::unique_ptr<T> b(len);
std::unique_ptr<T> c(len);
// 删除一个指针
// Delete a pointer
reset_ptr(a);
// 删除多个指针
// Delete multiple sequence pointers
reset_ptr(b, c);
```

bagrt::random_number

给定闭区间生成一个伪随机数

Generate a pseudo random number in given closed interval

参数 Parameters

double boundry_first

闭区间第一个值

First value of closed interval

double boundry_second

闭区间第二个值

Second value of closed interval

bool to_sleep

线程休眠信号

Thread sleep signal

double acc

精度

Accuracy

返回 Return

伪随机数

Pseudo random number

bagrt::input_paragraph

多个段落输入，双回车结束输入

Input paragraph, end by double pressing enter

参数 Parameters

uint64_t buffer_length

缓冲长度

Buffer length

返回 Return

段落字符串

Paragraph string

bagrt::extract_number

抽取一个字符串中的实数

Extract real numbers from a string

参数 Parameters

std::string num_str

目标字符串

Target string

返回 Return

实数数组指针

Real number array pointer

bagrt::charset_exchange

宽字符串与 ANSI 字符串互转

Transformation between wide string and ANSI string

参数 Parameters

ANSI 字符串转换为宽字符串

ANSI string transform to wide string

std::string str_src

目标 ANSI 字符串

Target ANSI string

返回 Return

对应宽字符串

Corresponding wide string

参数 Parameters

宽字符串转换为 ANSI 字符串

Wide string transform to ANSI string

std::wstring str_src

目标宽字符串

Target wide string

返回 Return

对应 ANSI 字符串

Corresponding ANSI string

bagrt::primes

获取给定上界的素数集合

Get prime sequence by given upper bound

参数 Parameters

uint64_t val

上界

Upper bound

返回 Return

素数集合

Prime set

bagrt::primes_fact

分解素数因数

Prime factor decomposition

参数 Parameters

uint64_t val

目标值

Target natural number

返回 Return

计算结果

Answer value

bagrt::greatest_common_divisor

获取两个给定自然数的最大公约数

Get greatest common divisor

参数 Parameters

uint64_t l_val

第一个数

First natural number

uint64_t r_val

第二个数

Second natural number

返回 Return

计算结果

Answer value

bagrt::least_common_multiple

获取两个给定自然数的最小公倍数

Get least common multiple

参数 Parameters

uint64_t l_val

第一个数

First natural number

uint64_t r_val

第二个数

Second natural number

返回 Return

计算结果

Answer value

csvio

头文件，CSV 文件输入输出.

Head file, CSV file I/O.

csvio::input_table

CSV 表格输入

Input a CSV table

参数 Parameters

std::string file_path

文件路径

File directory path

返回 Return

二维字符串顺序表

2D string sequence

csvio::output_table

CSV 表格输出

Output a CSV table

参数 Parameters

bagrt::net_queue<bagrt::net_queue> output_strings

目标二维表

Target 2D table

std::string file_path

保存文件路径

File save directory path

返回 Return

二维字符串顺序表

2D string sequence

netbaslib

头文件，神经网络基础库.

Head file, neural network basic library.

fc::Output

全连接前向传播

Fully connection forward propagation

参数 Parameters

vect vecInput

输入向量

Input vector

vect vecWeight

权重向量

Weight vector

返回 Return

前向传播输出

Forward propagation output

fc::GradLossToInput

全连接反向传播，获取损失到输入的梯度

Fully connection back propagation, get gradient from loss to input

参数 Parameters

vect vecGradLossToOutput

损失到输出梯度向量

Gradient vector from loss to output

vect vecWeight

权重向量矩阵

Weight vector matrix

返回 Return

反向传播到上一网络层的梯度向量

Gradient vector of back propagation to last network layer

fc::GradLossToWeight

全连接反向传播，获取损失到权重的梯度

Fully connection back propagation, get gradient from loss to weight

参数 Parameters

vect vecGradLossToOutput

损失到输出梯度向量

Gradient vector from loss to output

vect vecInput

输入向量

Input vector

返回 Return

用于更新权重的梯度向量

Gradient vector for updating weight

fc::FeatureTransform

向量张量互相变换

Vector and tensor transformation

参数 Parameters

张量转换为向量

Tensors transform to vectors

feature vecInput

输入张量

Input tensor

返回 Return

对应摊平变换向量

Corresponding flat transformed vector

参数 Parameters

向量转换为张量

vectors transform to tensors

vect vecInput

输入向量

Input vector

uint64_t iLnCnt

张量行计数

Tensor line count

uint64_t iColCnt

张量列计数

Tensor column count

返回 Return

对应转换张量

Corresponding transformed tensor

conv::Conv

卷积

Convolution

参数 Parameters

feature vecInput

输入张量，多通道矩阵

Input tensor, multi-channel matrices

tensor tenKernel

卷积核张量，多个多通道矩阵

Kernels tensor, multiple multi-channel matrices

uint64_t iLnStride

行向步幅

Line directed stride

uint64_t iColStride

列向步幅

Column directed stride

uint64_t iLnDilation

行向扩张

Line directed dilation

uint64_t iColDilation

列向扩张

Column directed dilation

uint64_t iInputPadTop

输入顶部扩展

Top padding for input

uint64_t iInputPadRight

输入右部扩展

Right padding for input

uint64_t iInputPadBottom

输入底部扩展

Bottom padding for input

uint64_t iInputPadLeft

输入左部扩展

Left padding for input

uint64_t iLnDistance

元素行向间距

Line directed distance between elements

uint64_t iColDistance

元素列向间距

Column directed distance between elements

返回 Return

前向传播卷积输出

Convolution output for forward propagation

conv::GradLossToKernel

获取损失到卷积核的梯度

Get gradient from loss to kernels

参数 Parameters

feature vecGradLossToOutput

损失到输出的梯度

Gradient from loss to output

feature vecInput

输入张量，多通道矩阵

Input tensor, multi-channel matrices

uint64_t iLnStride

行向步幅

Line directed stride

uint64_t iColStride

列向步幅

Column directed stride

uint64_t iLnDilation

行向扩张

Line directed dilation

uint64_t iColDilation

列向扩张
Column directed dilation
uint64_t iInputPadTop
输入顶部扩展
Top padding for input
uint64_t iInputPadRight
输入右部扩展
Right padding for input
uint64_t iInputPadBottom
输入底部扩展
Bottom padding for input
uint64_t iInputPadLeft
输入左部扩展
Left padding for input
uint64_t iLnDistance
元素行向间距
Line directed distance between elements
uint64_t iColDistance
元素列向间距
Column directed distance between elements
返回 Return
反向传播，更新卷积核的梯度
Back propagation, the gradient for updating kernels

conv::GradLossToInput
获取损失到输出的梯度
Get gradient from loss to input
参数 Parameters
feature vecGradLossToOutput
损失到输出的梯度
Gradient from loss to output
tensor tenKernel
卷积核张量，多个多通道矩阵
Kernels tensor, multiple multi-channel matrices
uint64_t iLnStride
行向步幅
Line directed stride
uint64_t iColStride
列向步幅
Column directed stride
uint64_t iLnDilation
行向扩张
Line directed dilation
uint64_t iColDilation
列向扩张
Column directed dilation

uint64_t iInputPadTop
输入顶部扩展
Top padding for input
uint64_t iInputPadRight
输入右部扩展
Right padding for input
uint64_t iInputPadBottom
输入底部扩展
Bottom padding for input
uint64_t iInputPadLeft
输入左部扩展
Left padding for input
uint64_t iLnDistance
元素行向间距
Line directed distance between elements
uint64_t iColDistance
元素列向间距
Column directed distance between elements
返回 Return
反向传播，损失到网络上一层激活输出的梯度
Back propagation, gradient from loss to last layer’s activation output

conv::PoolDown
下采样池化
Down-sample pooling
参数 Parameters
feature vecInput
输入张量，多通道矩阵，损失到池化下采样输出的梯度
Input tensor, multi-channel matrices, gradient from loss to down-sample pooling output
uint64_t iPoolType
池化类型
Pooling type
[POOL_DOWN_AVG] 平均下采样池化 Average down-sample pooling
[POOL_DOWN_MAX] 最大下采样池化 Max down-sample pooling
[POOL_DOWN_GAG] 全局平均下采样池化 Global average down-sample pooling
feature vecTraceInput
投影输入，下采样池化输入
Traced input, down-sample pooling input
uint64_t iFilterLnCnt
单位采样过滤行计数
Unit sample filter line count
uint64_t iFilterColCnt
单位采样过滤列计数
Unit sample filter column count
uint64_t iLnStride
行向步幅

Line directed stride

uint64_t iColStride

列向步幅

Column directed stride

uint64_t iLnDilation

行向扩张

Line directed dilation

uint64_t iColDilation

列向扩张

Column directed dilation

返回 Return

前向传播下采样池化输出

Down-sample pooling output for forward propagation

conv::PoolUp

上采样池化

Up-sample pooling

参数 Parameters

feature vecInput

输入张量，多通道矩阵

Input tensor, multi-channel matrices

uint64_t iPoolType

池化类型

Pooling type

[POOL_DOWN_AVG] 平均下采样池化 Average down-sample pooling

[POOL_DOWN_MAX] 最大下采样池化 Max down-sample pooling

[POOL_DOWN_GAG] 全局平均下采样池化 Global average down-sample pooling

uint64_t iFilterLnCnt

单位采样过滤行计数

Unit sample filter line count

uint64_t iFilterColCnt

单位采样过滤列计数

Unit sample filter column count

uint64_t iLnStride

行向步幅

Line directed stride

uint64_t iColStride

列向步幅

Column directed stride

uint64_t iLnDilation

行向扩张

Line directed dilation

uint64_t iColDilation

列向扩张

Column directed dilation

返回 Return

前向传播下采样池化输出

Down-sample pooling output for forward propagation

示例 Example

C++

```
// 前向传播使用最大下采样池化操作
// Using max down-sample pooling for forward propagation
auto vecPoolDownOutput = PoolDown(vecPoolInput, POOL_DOWN_MAX, 2, 2, 2, 2);
// 其他神经网络运算
// Other neural network operation
/* code */
/* 反向传输上采样池化
 * 传入之前池化的输入作为投影输入以及反向传回来的损失到池化输出梯度作为上采样池化的输入
 */
/* Up-sample pooling for back propagation
 * Input the previous pooling input as traced input and the back propagation
 * gradients from loss to pooling output as up-sample pooling input
 */
Auto vecGradLossToPoolDownInput = PoolUp(vecGradLossToPoolDownOutput,
                                         POOL_UP_MAX, vecPoolInput, 2, 2, 2, 2);
```

netbatlib

头文件，神经网络批量处理库.

这个库中包含了头文件 netbaslib 中神经网络基础函数的批量处理重载函数.

批量处理函数除了输入变量不同，其余参数均相同，故作省略.

Head file, neural network library for batch process.

This head file includes the overloaded functions of the basic neural network function in head file netbaslib for batch process.

Except the input variable difference, other parameters are same in batch process overloaded functions. The introduction is omitted.

fc::BNTrain

全连接批归一化，训练用

Fully connection batch normalization, for training

参数 Parameters

set<vect> setInput

输入向量集合

Input vector set

double dBeta

偏移度

Shift

double dGamma

尺度

Scale

double dEpsilon

除零占位

Zero devisor dominant

返回 Return

前向传播，全连接批归一化输出数据

Forward propagation, data structure of fully connection batch normalization output

fc::BNGradLossToInput

获取损失到批归一化输入的梯度

Get gradient from loss to batch normalization input

参数 Parameters

FCBN FCBNOutput

全连接批归一化输出数据结构

Data structure of fully connection batch normalization output

set<vect> setInput

输入向量集合

Input vector set

set<vect> setGradLossToOutput

损失到批归一化输出的梯度向量集合

Gradient vector set from loss to batch normalization output

double dGamma

尺度

Scale

double dEpsilon

除零占位

Zero devisor dominant

返回 Return

反向传播，损失到网络上一层的输出向量集合

Forward propagation, gradient vector set from loss to last layer’s output

fc::BNGradLossToScale

获取损失到尺度的梯度

Get gradient from loss to scale

参数 Parameters

set<vect> setGradLossToOutput

损失到批归一化输出的梯度向量集合

Gradient vector set from loss to batch normalization output

FCBN FCBNOutput

全连接批归一化输出数据结构

Data structure of fully connection batch normalization output

返回 Return

反向传播，用于更新尺度的梯度

Forward propagation, gradient for updating scale

fc::BNGradLossToSshift

获取损失到偏移度的梯度

Get gradient from loss to shift

参数 Parameters

set<vect> setGradLossToOutput

损失到批归一化输出的梯度向量集合

Gradient vector set from loss to batch normalization output

返回 Return

反向传播，用于更新偏倚度的梯度

Forward propagation, gradient for updating shift

fc::BNDeduce

全连接批归一化，推测用

Fully connection batch normalization, for deduction

参数 Parameters

set<vect> setNetInput

网络输入向量集合

Network input vector set

double dBeta

偏移度

Shift

double dGamma

尺度

Scale

set<BN_PTR> setbnData

所有批次的批归一化输出数据结构集合

All batches of batch normalization output data structure set

uint64_t iMiniBatchSize

每个批次的大小

Each mini-batch size

double dEpsilon

除零占位

Zero devisor dominant

返回 Return

用于推测的批归一化输出

Batch normalization output for deduction

conv::BNTrain

卷积批归一化，训练用

Convolution batch normalization, for training

参数 Parameters

set<feature> setInput

输入张量集合

Input tensor set

vect dBeta

偏移度向量

Shift vector

vect dGamma

尺度

Scale

double dEpsilon

除零占位

Zero devisor dominant

返回 Return

前向传播，卷积批归一化输出数据

Forward propagation, data structure of convolution batch normalization output

conv::BNGradLossToInput

获取损失到批归一化输入的梯度

Get gradient from loss to batch normalization input

参数 Parameters

ConvBN FCBNOutput

卷积批归一化输出数据结构

Data structure of convolution batch normalization output

set<feature> setInput

输入张量集合

Input tensor set

set<feature> setGradLossToOutput

损失到批归一化输出的梯度张量集合

Gradient tensor set from loss to batch normalization output

vect dGamma

尺度向量
Scale vector
double dEpsilon
除零占位
Zero devisor dominant
返回 Return
反向传播，损失到网络上一层的输出张量集合
Forward propagation, gradient tensor set from loss to last layer’s output

conv::BNGradLossToScale

获取损失到尺度的梯度
Get gradient from loss to scale
参数 Parameters
set<feature> setGradLossToOutput
损失到批归一化输出的梯度张量集合
Gradient tensor set from loss to batch normalization output
ConvBN ConvBNOutput
卷积批归一化输出数据结构
Data structure of convolution batch normalization output
返回 Return
反向传播，用于更新尺度的梯度
Forward propagation, gradient for updating scale

conv::BNGradLossToSshift

获取损失到偏移度的梯度
Get gradient from loss to shift
参数 Parameters
set<feature> setGradLossToOutput
损失到批归一化输出的梯度张量集合
Gradient tensor set from loss to batch normalization output
返回 Return
反向传播，用于更新偏移度的梯度
Forward propagation, gradient for updating shift

conv::BNDeduce

卷积批归一化，推测用
Convolution batch normalization, for deduction
参数 Parameters
set<feature> setNetInput
网络输入向量集合
Network input tensor set
vect dBeta
偏移度向量
Shift vector
vect dGamma
尺度向量

Scale vector

set<BN_PTR> setbnData

所有批次的批归一化输出数据结构集合

All batches of batch normalization output data structure set

uint64_t iMiniBatchSize

每个批次的大小

Each mini-batch size

double dEpsilon

除零占位

Zero devisor dominant

返回 Return

用于推测的批归一化输出

Batch normalization output for deduction

funclib

头文件，函数库．包括激活函数、工具函数等．

Head file, function library. Including activation functions and tool functions, etc.

sigmoid

sigmoid 激活函数

Activation function sigmoid

参数 Parameters

单个数值

Individual number

double val

变量

Variable

返回 Return

激活数值

Activation value

参数 Parameters

向量值

Vector value

vect vec_val

向量值

Vector value

返回 Return

激活向量

Activation vector

sigmoid_dv

sigmoid 导函数

Derivative function of sigmoid

参数 Parameters

单个数值

Individual number

double val

变量

Variable

返回 Return

激活梯度值

Activation gradient value

参数 Parameters

向量值

Vector value

vect vec_val

向量值

Vector value

返回 Return

激活梯度向量

Activation gradient vector

ReLU

ReLU 激活函数

Activation function ReLU

参数 Parameters

单个数值

Individual number

double val

变量

Variable

返回 Return

激活数值

Activation value

参数 Parameters

向量值

Vector value

vect vec_val

向量值

Vector value

返回 Return

激活向量

Activation vector

ReLU_dv

ReLu 导函数

Derivative function of ReLU

参数 Parameters

单个数值

Individual number

double val

变量

Variable

返回 Return

激活梯度值

Activation gradient value

参数 Parameters

向量值

Vector value

vect vec_val

向量值

Vector value

返回 Return

激活梯度向量

Activation gradient vector

softmax

softmax 激活函数

Activation function softmax

参数 Parameters

向量值

Vector value

vect vec_val

向量值

Vector value

返回 Return

激活向量

Activation vector

softmax_dv

softmax 导函数

Derivative function of softmax

参数 Parameters

vect vec_input

激活输入向量值

Input vector value for activation

vect vec_output

激活输出向量值

Activation output vector value

返回 Return

激活梯度向量

Activation gradient vector

cec_grad

交叉熵损失梯度

Cross-entropy cost gradient

参数 Parameters

向量

For vector

vect output

激活输出向量

Activation output vector

vect origin

标签分类向量

Classification vector of label

返回 Return

损失梯度向量

Gradient vector of loss

参数 Parameters

张量值

Tensor value

feature output

激活输出张量

Activation output tensor

feature origin

标签分类向量

Classification tensor of label

返回 Return

损失梯度张量

Gradient tensor of loss

参数 Parameters

张量集

Tensor set

set<feature> output

激活输出张量集

Activation output tensor set

set<feature> origin

标签分类向量

Classification tensor set of labels

返回 Return

损失梯度张量集

Gradient tensors set of loss

softmax_cec_grad

损失到 softmax 输入的梯度

Gradient from loss to softmax input

参数 Parameters

向量

For vector

vect softmax_output

softmax 输出向量

Output vector of softmax

vect origin

标签分类向量

Classification vector of label

返回 Return

损失到 softmax 输入的梯度向量

Gradient vector from loss to softmax input

参数 Parameters

向量集

For vector set

set<vect> softmax_output

softmax 输出向量集

Output vector set of softmax

set<vect> origin

标签分类向量集

Classification vector set of labels

返回 Return

损失到 softmax 输入的梯度向量集

Gradient vector set from loss to softmax input

参数 Parameters

张量

For tensor

tensor softmax_output

softmax 输出张量

Output tensor of softmax

tensor origin

标签分类张量

Classification tensor of labels

返回 Return

损失到 softmax 输入的梯度张量

Gradient tensor from loss to softmax input

命令 Command

这个小节将介绍几个宏定义命令，这些命令可以用来辅助程序运行以及监视。命令需要包含了定义所在的头文件才能使用，这几个命令是全局性的。

This chapter will introduce some commands which could assist program running and monitoring. Commands need include its definition file to use. This several commands are global.

funclib

INSTANCE_DERIVE

DIV_DOM

SAMP_BLOCK_CNT

SAMP_TRACE_POS

SAMP_OUTPUT_DIR_CNT

SAMP_INPUT_DIR_CNT

SAMP_VALID

bagrt

CLOCK_BEGIN

CLOCK_END

CLOCK_DURATION

funclib

头文件，函数库.

Head file, function library.

INSTANCE_DERIVE

实例派生转换

Instance derivation transformation

模板 Template

_Ty1

子类

Derivative class

参数 Parameters

shared_ptr _Other

多态父类对象指针

Polymorphic parent instance pointer

返回 Return

子类对象指针

Derivative instance pointer

DIV_DOM

向量除零占位处理，将矩阵中为 0 的元素使用占位替换

Zero divisor dominant process for vector, supersede the 0 element with dominant in matrix

参数 Parameters

vect divisor

目标除数向量

Target divisor vector

double epsilon

除零占位值

Zero divisor dominant

返回 Return

非零除数矩阵

Matrix divisor without 0 element

SAMP_BLOCK_CNT

采样块尺寸计数

Sample black size count

参数 Parameters

uint64_t filter_dir_cnt

过滤方向计数

Filter direction count

uint64_t dir_dilation

指定方向扩张值

Dilation of appointed direction

返回 Return

尺寸大小

Size count

示例 Example

C++

```
// 下采样池化中，计算长宽为 2，长宽元素扩张值为 1 的采样块大小
/* In down-sample pooling, calculate the sample block size with 2 units long line & column
 * and 1 unit line & column dilation
 */
auto iFilterLnCnt = SAMPLE_BLOCK_CNT(2, 1),
    iFilterColCnt = SMAPLE_BLOCK_CNT(2, 1);
```

SAMP_TRACE_POS

采样输出的元素位置对输入的投影位置

Element position of sample input traced from output element position

参数 Parameters

uint64_t output_dir_pos

输出元素的目标方向位置

Position of output element in target direction

uint64_t filter_dir_pos

过滤目标方向位置

Position of filter in target direction

uint64_t dir_stride

目标方向步幅

Stride of target direction

uint64_t dir_dilation

目标方向元素间扩张值

Dilation between elements of target direction

返回 Return

投影位置

Position from trace

示例 Example

C++

```
// 在步幅为 2，扩张值为 1 的卷积计算中，求输出向量(2, 3)位置的元素卷积
 * 由卷积核(0, 1)位置的元素和输入向量那个位置的向量元素的乘积求和得到
/* In convolution calculation with 2 strides and 1 dilation, Find the position of input
 * vector which calculate the sum for convolution of output vector element in position
 * (2, 3) with the product between kernel element in position (0, 1)
 * and 1 unit line & column dilation
 */
auto iTargetInputLn = SAMPLE_TRACE_POS(2, 0, 2, 1),
    iTargetInputCol = SAMPLE_TRACE_POS(3, 1, 2, 1);
```

SAMP_OUTPUT_DIR_CNT

获取采样输出目标方向的计数

Get sample output target direction count

参数 Parameters

uint64_t intput_dir_cnt

输入向量的目标方向计数

Input vector target direction count

uint64_t filter_dir_cnt

过滤目标方向计数

Filter target direction count

uint64_t dir_stride

目标方向步幅

Stride of target direction

uint64_t dir_dilation

目标方向元素间扩张值

Dilation between elements of target direction

返回 Return

输出向量目标方向计数值

Target direction count value of output vector

示例 Example

C++

```
// 在步幅为 2，扩张值为 1 的卷积计算中，输入向量行列为 6，卷积核行列为 2，求输出向量行列数值
/* In convolution calculation with 2 strides and 1 dilation, Find the line & column count
 * of output vector with 6 units of input line & column count and 2 units of kernel line
 * & column count
 */
auto iOutputLnCnt = SAMPLE_OUTPUT_DIR_CNT(6, 2, 2, 1),
    iOutputColCnt = SAMPLE_OUTPUT_DIR_CNT(6, 2, 2, 1);
```

SAMP_INPUT_DIR_CNT

获取采样输入目标方向的计数

Get sample input target direction count

参数 Parameters

uint64_t output_dir_cnt

输出向量的目标方向计数

Output vector target direction count

uint64_t filter_dir_cnt

过滤目标方向计数

Filter target direction count

uint64_t dir_stride

目标方向步幅

Stride of target direction

uint64_t dir_dilation

目标方向元素间扩张值

Dilation between elements of target direction

返回 Return

输入向量目标方向计数值

Target direction count value of input vector

SAMP_VALID

验证采样是否可行

Sampling viable validation

参数 Parameters

uint64_t intput_dir_cnt

输入向量的目标方向计数

Input vector target direction count

uint64_t filter_dir_cnt

过滤目标方向计数

Filter target direction count

uint64_t dir_stride

目标方向步幅

Stride of target direction

uint64_t dir_dilation

目标方向元素间扩张值

Dilation between elements of target direction

返回 Return

验证布尔值

Validation Boolean value

bagrt

头文件，基础算法库.

Head file, basic algorithm library.

CLOCK_BEGIN

时钟开始点，设置于目标程序段开始之前

Clock point begin, before the beginning of the target program segment

警告 Warning

该命令必须与命令 CLOCK_END 成对出现并在之前，不可重复

This command must be a pair with command CLOCK_END and before it, unduplicable.

CLOCK_END

时钟结束点，设置于目标程序段结束之后

Clock point end, after the end of the target program segment

警告 Warning

该命令必须与命令 CLOCK_BEGIN 成对出现并在之后，不可重复

This command must be a pair with command CLOCK_BEGIN and after it, unduplicable.

CLOCK_DURATION

时钟长度，用于获取计时命令之间程序段运行的时间

Clock duration, get duration of the program segment between the clock point commands.

示例 Example

C++

```
// 获取从 1 循环加到 100 的时间
// Get the time cost of 1 adding to 100
CLOCK_BEGIN
auto sum = 1;
while (++i <= 100);
CLOCK_END
std::cout << CLOCK_DURATION << std::endl;
```

类 Class

这个小节将介绍几个常用的类。这些类包括线性表、矩阵、数据集以及位图。介绍包括可访问的字段、成员函数以及运算符。这些类将通过所属的头文件进行介绍。派生将在末尾列出。

This chapter will introduce some common class. These classes are linear sequence, matrix, dataset, and bitmap. Introduction includes the accessible fields, member functions and operator. The introduction of these classes will follow their head file. Derivations will be listed in the end.

bagrt
bagrt::net_queue
bagrt::net_list
bagrt::net_map
matrix
mtx::matrix
bmio
bmio::bitmap
dataset
dataset::MNIST
neunet
neunet::NetBase
neunet::NetClassify

bagrt

头文件，基础算法库.

Head file, basic algorithm library.

bagrt::net_queue

顺序表

Sequence

常量 Constant

set

vect_t = bagrt::net_queue<bagrt::net_queue>

模板 template

_Ty

列存储元素类型

Type name of the element stored in sequence

函数 Function

net_queue

构造函数

Constructor

参数 Parameter

uint64_t _size

分配内存长度

Length of memory allocation

blank_queue

返回一个空列

Return a blank sequence

常量 Constant

blank_feature

空张量

Blank tensor

空张量集合

Blank tensor set

blank_tensor

空张量结合列

Blank tensor set sequence

blank_vect_seq

空向量列

Blank vector sequence

blank_ft_seq

空张量列

Blank tensor sequence

blank_ten_seq

空张量集合列

Blank tensor set sequence

size

返回内存长度

Return size of memory length

init

初始化

Initialization

参数 Parameter

uint64_t _size

分配内存长度

size of memory allocation

返回 Return

初始化成功验证布尔值

Successfully initialization validation Boolean value

insert

插入一个元素

Insert an element

参数 Parameter

uint64_t idx

插入位置

Index of inserting

Args ... args

用于元素构造的参数

Parameter for element construction

返回 Return

插入成功验证布尔值

Successfully insert validation Boolean value

示例 Example

C++

```
// 声明一个列变量并分配 1 个内存，模板为 matrix
// Declare a sequence variable and allocate 1 unit memory, template is matrix
bagert::net_queue<mtx::matrix> test(1);
// [test.init();] or [test.init(1);]
// 第一个位置赋值
// Assign the first index element
test[0] = {{1, 2},
           {3, 4}};
// 在第 2 个位置插入一个值
// Insert a value at second index
test.insert(1, {{2, 4},
                {6, 8}});
/* 当然也可以先声明一个矩阵再插入
 * It can also declare a matrix instance before inserting
 * mtx::matrix temp = {{2, 4},
 *                     {6, 8}};
 * test.insert(1, temp);
 * 这两者效果相同
 * These two have the same effect
 * 此时的列表内存长度为 2
 * The sequence size is 2 now
```

```
 */
 std::cout << test.size() << std::endl;
 // 将打印出 2
 // Will print number 2
 std::cout << test << std::endl;
 /* 打印结果
  * Print result
  * [0][
  * 1 2
  * 3 4
  * ]
  * [1][
  * 2 4
  * 6 8
  * ]
  */
```

emplace_back

向列末尾添加一个元素

Add an element at the end of sequence

参数 Parameter

Args ... args

用于元素构造的参数

Parameter for element construction

返回 Return

添加成功验证布尔值

Successfully add validation Boolean value

push_back

向列表末尾添加一个元素

Add an element at the end of sequence

参数 Parameter

_Ty val

新元素

New element

返回 Return

添加成功验证布尔值

Successfully add validation Boolean value

erase

擦除多个列表元素

Erase several elements from sequence

参数 Parameter

Args ... idx

多个元素的位置

Multi-index of elements

返回 Return

被擦除元素列

Elements sequence erased

示例 Example

C++

```
// 声明一个列变量并分配 5 个内存，模板为 double
// Declare a sequence variable and allocate 1 unit memory, template is double
bagrt::net_queue<double> test(5);
// 迭代填充
// Filled by iteration
for(auto i=0; i<test.size(); ++i) test[i] = i;
// 擦除
// Erase
std::cout << test.erase(2) << '\n' << std::endl;
std::cout << test << '\n' << std::endl;
std::cout << test.erase(1, 2, 3) << '\n' << std::endl;
std::cout << test << std::endl;
/* 打印结果
 * Print result
 * [0][
 * 2
 *]
 *
 * [0][
 * 0
 * ]
 * [1][
 * 2
 * ]
 * [2][
 * 3
 * ]
 * [3][
 * 4
 * ]
 * [4][
 * 5
 * ]
 *
 * [0][
 * 2
 * ]
 * [1][
 * 3
 * ]
 * [2][
 * 4
 *
```

 * [0][
 * 0
 *]
 * [1][
 * 5
 */

sub_queue

根据两个位置点获取子列

Get child sequence by two indices

参数 Parameter

uint64_t idx_first

第一个位置

First index

uint64_t idx_second

第二个位置

Second index

返回 Return

子列

Child sequence

sort

列元素排序，具体示例可以参照函数 bagrt::quick_sort

Sort the sequence element, for details referring to function bagrt::quick_sort

参数 Parameter

bool asc

升序信号

Ascent signal

std::function _func

升序定义函数

Ascent definition function

unit

联合另一个列

Unit another sequence

参数 Parameter

net_queue val

联合源

Source sequence for joint

返回 Return

联合列

United sequence

unit_union

与另一个列的并集

参数 Parameter

net_queue val

源

Source sequence

返回 Return

并集

Union sequence

unit_intersect

与另一个列的交集

参数 Parameter

net_queue val

源

Source sequence

返回 Return

交集

Intersection sequence

find

寻找列某个区间内的某个元素

Find an element in a specific range of sequence

参数 Parameter

_Ty target

目标

Target

uint64_t range_first

范围的第一个界

First bound of the range

uint64_t range_second

范围的第二个界

Second bound of the range

返回 Return

目标值所在位置集合

Index set the target located

sum

列元素求和

Get sum of sequence elements

参数 Parameter

std::function add_func

求和函数定义

Function definition of sum

返回 Return

求和结果

Answer of sum

reset

重置列

Reset sequence

运算符 Operator

[]

取列对应位置的引用值

Get the quoted value corresponding to the index in sequence

==

与另外一个列判定是否相同

Judge if same as another sequence

!=

与另外一个列判定是否不同

Judge if different from another sequence

bagrt::net_list

链表

List

常量 Constant

NET_LIST

模板 template

_Ty

列存储元素类型

Type name of the element stored in sequence

函数 Function

empty

判定是否为空

Judge if empty the list

size

返回链表长度

Return length of the list

Insert

特定位置插入一个元素

Insert an element at specific index

参数 Parameter

uint64_t idx

插入位置

Index of inserting

Args ... src

用于元素构造的参数

Parameter for element construction

返回 Return

插入成功验证布尔值

Successfully insert validation Boolean value

emplace_back

向表末尾添加一个元素

Add an element at the end of list

参数 Parameter

Args ... args

用于元素构造的参数

Parameter for element construction

返回 Return

添加成功验证布尔值

Successfully add validation Boolean value

erase

擦除一个目标位置的元素

参数 Parameter

uint64_t idx

目标位置

Target index

返回 Return

被擦除元素

The erased element

unit

联合另一个表

Unit another list

参数 Parameter

net_list val

联合源

Source sequence for joint

返回 Return

联合表

United list

unit_union

与另一个表并集

参数 Parameter

net_list val

源

Source list

返回 Return

并集

Union list

unit_intersect

与另一个列的交集

参数 Parameter

net_list val

源

Source list

返回 Return

交集

Intersection list

reset

重置表

Reset list

运算符 Operator

[]

取表对应位置的引用值

Get the quoted value corresponding to the index in list

==

与另外一个列判定是否相同

Judge if same as another list

!=

与另外一个列判定是否不同

Judge if different from another list

bagrt::net_map

键值表

Map

常量 Constant

NET_MAP

模板 template

_K

键类型

Key type name

_V

值类型

Value type name

函数 Function

size

返回键值表长度

Return length of the map

find_idx

寻找目标键在表中的位置

Find the index of target key in map

参数 Parameter

_K key

目标键

Target key

返回 Return

目标键的所在位置，如果不存在该键则返回-1

The index of target key, return -1 as target key does not exist

find_key

根据目标值寻找对应键

Find corresponding keys by target value

参数 Parameter

_V value

目标值

Target value

返回 Return

指向目标值的所有键列

The sequence of the keys pointed at target value

insert

插入一个键值对

Insert a key-value couple

参数 Parameter

_K key

目标键

Target key

_V value

目标值

Target value

返回 Return

成功插入验证布尔值

Successfully insert Boolean value validation

index

通过键值表位置获得对应键值对

Get corresponding key-value couple by index in map

参数 Parameter

uint64_t idx

目标位置

Target index

返回 Return

目标键值对引用

Quoted value of target key-value couple

reset

重置键值表

Reset map

运算符 Operator

[]

通过键获取对应键值对的值

Get the corresponding value of a key-value couple by key

==

与另外一个键值表判定是否相同

Judge if same as another map

!=

与另外一个键值表判定是否不同

Judge if different from another map

matrix

头文件，矩阵库.

由于矩阵已经封装成类,头文件函数介绍省略.

Head file, matrix library.

Since the matrix has been encapsulated as class, the introduction of functions in head file are omitted.

mtx::matrix

矩阵

Matrix

常量 Constant

vect

feature = bagrt::net_queue<mtx::matrix>

tensor = bagrt::net_queue<bagrt::net_queue<mtx::matrix>>

函数 Function

blank_matrix

返回空矩阵

Return blank matrix

常量 Constant

blank_vect

is_matrix

矩阵存在判定

Matrix existence validation

ptr

获取矩阵指针

Get matrix pointer

matrix

构造函数

Constructor

参数 Parameter

使用行列构造矩阵，选择是否使用随机数进行初始化

Use line & column count to construct, the pseudo random number initialization is selectable

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

bool rand

随机数初始化信号

Pseudo random number initialization signal

double rand_boundry_first

随机数区间第一个值

First value of pseudo random number initialization interval

double rand_boundry_second

随机数区间第二个值

Second value of pseudo random number initialization interval

double rand_acc

随机数精度

Pseudo random number accuracy

参数 Parameter

使用矩阵指针移动构造矩阵

Use matrix pointer to move construct a matrix

MATRIX ptr_val

矩阵指针

Matrix pointer

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

参数 Parameter

使用矩阵指针复制构造矩阵

Use matrix pointer to copy construct a matrix

MATRIX ptr_val

矩阵指针

Matrix pointer

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

参数 Parameter

原子矩阵

Atomic matrix

double atom

原子值

Atomic value

参数 Parameter

直接构造矩阵

Construct matrix directly

std::initializer_list<std::initializer_list<double>> _vect

矩阵初始化列

Matrix initializer list

示例 Example

```
matrix vector = {{1, 2},
                 {3, 4}};
std::cout << vector << std::endl;
/* 打印结果
 * Print result
```

 * 1 2
 * 3 4
 */

value_fill

使用目标值填充一个矩阵

Fill the matrix with target value

double val

目标值

get_ln_cnt

行计数

Line count

get_col_cnt

列计数

Column count

get_elem_cnt

元素个数

Element count

determinant

行列值

Determinant

inverser

逆矩阵

Inverser

transposition

转置

Transposition

extremum

寻找扩张式子矩阵最值

Find extremum from dilation matrix

参数 Parameter

uint64_t from_ln

行始值

Line beginning index

uint64_t to_ln

行尾值

Line end index

uint64_t from_col

列始值

Column beginning index

uint64_t to_col

列尾值

Column end index

uint64_t ln_dilation

行扩张值

Line dilation

uint64_t col_dilation

列扩张值

Column dilation

bool max_flag

最大值信号

Max value signal

返回 Return

矩阵元素极值数据结构

Data structure of matrix element extremum

示例 Example

C++

```
/*         1 4 2
 *寻找矩阵  2 4 0 的最值和坐标
 *         3 1 0
 * Get extremum from the matrix and its position in matrix
 */
mtx::matrix vector = {{1, 4, 2},
                      {2, 4, 0},
                      {3, 1, 0}};
auto vector_max = vector.extremum(0, 2, 0, 2);
auto vector_min = vector.extremum(0, 2, 0, 2, 0, 0, false);
std::cout << “[Max value][” << vector_max.val << ‘]’ << std::endl;
std::cout << “[Max position]” << std::endl;
std::cout << vector_max.pos_list << std::endl;
std::cout << “[Min value][” << vector_min.val << ‘]’ << std::endl;
std::cout << “[Min position]” << std::endl;
std::cout << vector_min.pos_list << std::endl;
/* 打印结果
 * Print result
 * [Max value][4]
 * [Max position]
 * [0][
 * (1, 2)
 * ]
 * [1][
 * (2, 2)
 * ]
 * [Min value][0]
 * [Min position]
 * [0][
 * (0, 1)
 * ]
 * [1][
 * (1, 1)
 * ]
 */
// 获取矩阵扩张值为 1 子矩阵元素的最大值
```

```
// Get the max value of dilation child matrix elements within 1 dilation
auto vector_dilation_max = vector.extremum(0, 2, 0, 2, 1, 1);
std::cout << "[Max dilation value][" << vector_dilation_max.val << ']' << std::endl;
std::cout << "[Max dilation position]" << std::endl;
std::cout << vector_dilation_max.pos_list << std::endl;
/* 打印结果
 * Print result
 * [Max dilation value][3]
 * [Max dilation position]
 * [0][
 * (2, 0)
 * ]
 */
```

matrix child

获取扩张式子矩阵

Child matrix from dilation matrix

参数 Parameter

uint64_t from_ln

行始值

Line beginning index

uint64_t to_ln

行尾值

Line end index

uint64_t from_col

列始值

Column beginning index

uint64_t to_col

列尾值

Column end index

uint64_t ln_dilation

行扩张值

Line dilation

uint64_t col_dilation

列扩张值

Column dilation

返回 Return

矩阵元素极值数据结构

rotate_rect

顺时针或逆时针直角旋转矩阵

Clockwise or anti-clockwise

参数 Parameter

bool clockwise

顺时针信号

Clockwise signal

返回 Return

旋转结果

Rotation result

mirror_flip

垂直或水平镜面反转矩阵元素

Vertical or horizon mirror flip the matrix element

参数 Parameter

bool is_vertical

垂直信号

Vertical signal

返回 Return

翻转结果

Flipping result

shape_valid

矩阵尺寸验证

Matrix shape validation

参数 Parameter

测试矩阵是否符合某个尺寸

Test if the matrix shape as a specific size

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

返回 Return

行列数值形状验证

Line and column value shape validation

参数 Parameter

测试矩阵是否符合另一个矩阵的形状

Test if matrix shape as another matrix’s shape

matrix mtx_src

对比源

Compare source

返回 Return

特定矩阵变量形状验证

Specific matrix variable shape validation

reshape

保持矩阵元素个数更改矩阵尺寸

Reshape the matrix without the element amount changing

参数 Parameter

指定重置尺寸参数

Appoint the reshape parameter

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数
Column count
返回 Return
重置矩阵，行列乘积必须保持不变
Reshaped matrix, the product of line & column count must be no changed
参数 Parameter
指定重置模板矩阵
Appoint the model matrix
matrix mtx_src
模板矩阵
Model matrix
返回 Return
重置矩阵，行列乘积必须保持不变
Reshaped matrix, the product of line & column count must be no changed
elem_sum
对矩阵元素求和
Get sum of the matrix elements
默认空参数列
Default blank parameter
返回 Return
所有元素求和结果
Sum of matrix all elements
参数 Parameter
对扩张子矩阵的元素求和
Get sum of the dilation child matrix elements
uint64_t from_ln
行始值
Line beginning index
uint64_t to_ln
行尾值
Line end index
uint64_t from_col
列始值
Column beginning index
uint64_t to_col
列尾值
Column end index
uint64_t ln_dilation
行扩张值
Line dilation
uint64_t col_dilation
列扩张值
返回 Return
子矩阵求和结果
Sum of child matrix

abs

返回 Return

绝对值矩阵

Absolute value matrix

elem_cal_opt

参数 Parameter

矩阵元素乘除运算

Multiplication and division (between the whole elements of the both matrices) calculation between matrix elements

matrix r_val

运算右矩阵

Right matrix for calculation

uint64_t opt_idx

运算参数

Operation index

[MATRIX_ELEM_MULT] 哈达玛积(元素相乘) Hadamard product(Element production)

[MATRIX_ELEM_DIV] 元素作商 Element quotient

返回 Return

计算结果

Answer result

参数 Parameter

矩阵元素乘方或商(值与矩阵元素之间)运算

Power and division (between a value and matrix elements) calculation of matrix elements

double para

运算值

Calculation parameter

uint64_t opt_idx

运算参数

Operation index

[MATRIX_ELEM_POW] 元素乘方 Power calculation of elements

[MATRIX_ELEM_DIV] 元素作商 Quotient between

返回 Return

计算结果

Answer result

示例 Example

C++

```
/*            1 2 3            0 2 4
 * 设定矩阵 A = 4 5 6 和矩阵 B = 6  8 10
 *             7 8 9           12 14 16
 * Set matrix A and B
 */
mtx::matrix A = {{1, 2, 3},
                 {4, 5, 6},
                 {7, 8, 9}};
mtx::matrix B = {{ 0, 2, 4},
```

```
                    { 6, 8,10},
                    {12,14,16}};
/* 计算 A 与 B 元素的之间的商(A 元素为除数)，乘积
 *      A 元素的 1.5 次方矩阵
 *      B 矩阵元素和 2.2 作商的矩阵
 * Calculate The quotient (A as divisor) and product of the element in A and B respectively
 *           The matrix filled by the element result from element of A by power 1.5 times
 *         The matrix filled by the element result from element of B by quotient with 2.2
 */
std::cout << B.elem_cal_opt(A, MATRIX_ELEM_DIV) << std::endl << std::endl;
std::cout << A.elem_cal_opt(B, MATRIX_ELEM_MULT) << std::endl << std::endl;
std::cout << A.elem_cal_opt(1.5, MATRIX_ELEM_POW) << std::endl << std::endl;
std::cout << A.elem_cal_opt(2.2, MATRIX_ELEM_DIV) << std::endl;
/* 打印结果
 * Print result
 * 0       1       1.33333
 * 1.5     1.6     1.66667
 * 1.71429 1.75    1.77778
 *
 * 0       4       12
 * 24      40      60
 * 84      112     144
 *
 * 1       2.82843 5.19615
 * 8       11.1803 14.6969
 * 18.5203 22.6274 27
 *
 * 0.454545 0.909091 1.36364
 * 1.81818  2.27273  2.72727
 * 3.18182  3.63636  4.09091
 */
```

pad

以 0 扩展矩阵四周以及元素距离

Padding the matrix four sides and element distance with 0

参数 Parameter

uint64_t ln_t

行顶

Line top

uint64_t col_r

列右

Column right

uint64_t ln_b

行底

Line bottom

uint64_t col_l

列左

Column left

uint64_t ln_dist

行间

Line distance

uint64_t col_dist

列间

Column distance

返回 Return

扩展矩阵

Padded matrix

示例 Example

C++

```
/*         1 4 2
 * 扩展矩阵 2 4 0 上下 2 个单位，左右 1 个单位，元素间 1 个单位
 *         3 1 0
 * Padding the matrix element 2 units 1, 1 unit in top & bottom, left & right,
 * elements distance respectively
 */
mtx::matrix vector = {{1, 4, 2},
                      {2, 4, 0},
                      {3, 1, 0}};
std::cout << vector.pad(2, 1, 2, 1, 1, 1) << std::endl;
/* 打印结果
 * Print result
 * 0 0 0 0 0 0 0
 * 0 0 0 0 0 0 0
 * 0 1 0 4 0 2 0
 * 0 0 0 0 0 0 0
 * 0 2 0 4 0 0 0
 * 0 0 0 0 0 0 0
 * 0 3 0 1 0 0 0
 * 0 0 0 0 0 0 0
 * 0 0 0 0 0 0 0
 */
```

crop

裁切四周即元素间的矩阵元素

Cropping the elements four sides and between the elements of the matrix

参数 Parameter

uint64_t ln_t

行顶

Line top

uint64_t col_r

列右

Column right

uint64_t ln_b

行底

Line bottom

uint64_t col_l

列左

Column left

uint64_t ln_dist

行间

Line distance

uint64_t col_dist

列间

Column distance

返回 Return

Cropped matrix

裁剪矩阵

示例 Example

C++

```
/*         0 1 4 9 3 0
 *         1 4 2 4 7 14
 *         1 9 1 5 7 6
 * 裁切矩阵 2 4 0 9 6 8 四周 1 个单位，行向元素间 1 个单位，列向元素间 2 个单位
 *         3 1 0 1 8 4
 *         0 9 2 8 6 4
 *         2 0 1 6 7 5
 * Cropping matrix 1, 1 unit, 2 units at four side, line & column directed
 * elements distance respectively
 */
mtx::matrix vector = {{0, 1, 4, 9, 3, 0},
                      {1, 4, 2, 4, 7, 14},
                      {1, 9, 1, 5, 7, 6},
                      {2, 4, 0, 9, 6, 8},
                      {3, 1, 0, 1, 8, 4},
                      {0, 9, 2, 8, 6, 4},
                      {2, 0, 1, 6, 7, 5}};
std::cout << vector.crop(1, 1, 1, 1, 1, 2) << std::endl;
/* 打印结果
 * Print result
 * 4 7
 * 4 6
 * 9 6
 */
```

round_fit

返回 Return

四舍五入元素矩阵

Matrix filled by element rounded value

matrix LU()

返回 Return

LU 分解组合矩阵

LU decomposition combination matrix

linear_eq

解线性方程组

Answer the system of linear equations

参数 Parameter

matrix val_b

右侧向量

Right side vector

bool eq_idx

方程组解法

Solution index

[MATRIX_EQ_LU]　　LU 分解 LU decomposition

[MATRIX_EQ_JACOBI]　Jacobi 迭代 Jacobi iteration

返回 Return

特解矩阵

Particular solution matrix

swap_dir_elem

初等行列变换，两个行列值交换

Primary row & column transformation, exchange all elements between two line & column

参数 Parameter

uint64_t l_idx

第一个交换位置

First exchange index

uint64_t r_idx

第二个交换位置

Second exchange index

bool is_ln

行变换信号

Line transformation signal

返回 Return

变换矩阵

Transformation matrix

adjugate

计算伴随矩阵

Get adjugated matrix

参数 Parameter

uint64_t ln

目标行

Target line

uint64_t col

目标列

Target column

返回 Return

矩阵目标位置伴随矩阵

The adjugated matrix of the target position in matrix

rank

返回 Return

Matrix rank

矩阵秩

pos_idx

获取矩阵指针单维位置值

Get value of single dimension matrix pointer at target index

参数 Parameter

uint64_t idx

指针位置

Pointer index

返回 Return

矩阵指针位置元素引用

Quote value of the matrix pointer at target index

reset

重置矩阵

Reset matrix

运算符 Operator

[][]

取矩阵元素

Get matrix element

+ +=

矩阵加法

Matrix addition

- -=

矩阵减法

Matrix subtraction

*** *=**

矩阵乘法

Matrix multiplication

==

判定矩阵是否相同

Judge if matrix same as another one

!=

判定矩阵是否不同

Judge if matrix different from another one

bmio

头文件，位图库.

Head file, bitmap library.

bmio::bitmap

位图输入输出处理类，可以根据文件位置路径输入图片以矩阵形式进行修改并保存到指定文件位置.

Bitmap I/O & process class. It can input the image according to the file directory path, process by matrix form and save at the appointed file directory.

函数 Function

bitmap

构造函数

Constructor

BMIO_RAW = bagrt::net_queue<mtx::matrix>

BMIO_STR = std::string

BMIO_STR = std::wstring

参数 Parameter

使用向量集合进行移动构造

Use vector set to move construct

BMIO_RAW vec

图片向量集合

Image vector set

参数 Parameter

使用向量集合进行复制构造

Use vector set to move construct

BMIO_RAW vec

图片向量集合

Image vector set

参数 Parameter

使用图片所在路径输入图片

Use the image directory path to input

BMIO_STR dir

图片路径，ANSI 字符串

Image file directory path, ANSI string

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

参数 Parameter

使用图片所在路径输入图片

BMIO_WSTR dir

图片路径，宽字符串

Image file directory path, wide string

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

set_size

设定图片尺寸

Set image size

参数 Parameter

uint64_t ln_cnt

行计数

Line count

uint64_t col_cnt

列计数

Column count

返回 Return

成功设定验证

Successfully select validation

set_raw

设定图片通道矩阵

Set image raw vectors for channels

BMIO_CHANN = mtx::matrix

参数 Parameter

BMIO_CHANN R_src

红色通道矩阵

Vector for red channel

BMIO_CHANN G_src

绿色通道矩阵

Vector for green channel

BMIO_CHANN B_src

蓝色通道矩阵

Vector for blue channel

BMIO_CHANN A_src

Alpha 通道矩阵

Vector for Alpha channel

bool move_flag

移动赋值信号

Move assignment signal

返回 Return

成功设定验证

Successfully select validation

ln_cnt

图片尺寸行计数

Line directed count of image size

col_cnt

图片尺寸列计数

Column directed count of image size

img_valid

图片验证

Image validation

load_img

使用图片文件目录路径载入图片

Load image by image file directory path

参数 Parameter

宽字符串

Wide string

BMIO_WSTR dir

图片路径，宽字符串

Image file directory path, wide string

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

返回 Return

成功加载验证

Successfully load validation

参数 Parameter

ANSI 字符串

ANSI string

BMIO_WSTR dir

图片路径，ANSI 字符串

Image file directory path, ANSI string

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

返回 Return

成功加载验证

Successfully load validation

save_img

将 RGB(A)矩阵以特定格式图片保存到指定文件目录

Save RGB(A) matrix at target file directory as image in specific format

参数 Parameter

ANSI 字符串

ANSI string

BMIO_STR dir_root

目标根目录

Target directory root

BMIO_STR name

图片文件名称

Image file name

uint64_t extend

扩展名，格式

Extend, format

[BMIO_PNG][BMIO_JPG][BMIO_GIF][BMIO_TIF][BMIO_BMP]

支持 Alpha 通道的格式

Formats support Alpha channel

[BMIO_PNG][BMIO_TIF]

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

char div_syb

路径结点分隔符

Separator of path node

['\\'] 转义反斜线 Escape backslash

['/'] 正斜线 Slash

返回 Return

成功保存验证

Successfully save validation

参数 Parameter

宽字符串

Wide string

BMIO_WSTR dir_root

目标根目录

Target directory root

BMIO_WSTR name

图片文件名称

Image file name

uint64_t extend

扩展名，格式

Extend, format

[BMIO_PNG][BMIO_JPG][BMIO_GIF][BMIO_TIF][BMIO_BMP]

支持 Alpha 通道的格式

Formats support Alpha channel

[BMIO_PNG][BMIO_TIF]

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

wchar_t div_syb

路径结点分隔符

Separator of path node

[L'\\'] 转义反斜线 Escape backslash

[L'/'] 正斜线 Slash

返回 Return

成功保存验证

Successfully save validation

gray

使用 Luma 码进行灰度化，CCIR 601 标准，静态函数

Grayscle by Luma codes, CCIR 601 standard, static function

返回 Return

Luma 码加权灰度矩阵

Weighted matrix by Luma codes

gray_grad

灰度梯度，静态函数

Gradient of grayscale, static function

参数 Parameter

BMIO_CHANN grad_vec

灰度向量梯度

Gradient of grayscale vector

返回 Return

RGB 通道梯度矩阵

Gradient vector of RGB channels

img_vec

获取图片所有通道的矩阵

Get vectors of all image channels

参数 Parameter

bool rgba

RGBA 信号，为真时添加 Alpha 通道，为假时默认 RGB

RGBA signal, increase Alpha channel for true value, default RGB for false value

返回 Return

通道矩阵集合

Vector set of channels

reset

重置图片

Reset image

运算符 Operator

==

判定图片的像素值，尺寸是否和另外一个图片相同

Judge if the image pixel value and size is same as another one

!=

判定图片的像素值，尺寸是否和另外一个图片不同

Judge if the image pixel value and size is different from another one

dataset

头文件，数据集运行库.

Head file, dataset run library.

dataset::MNIST

MNIST 数据集输入输出处理类． 可按照文件位置路径输入 MNIST 数据以矩阵方式进行处理并以图片形式保存到指定目录路径.

MNIST dataset I/O & process class. It can input the MNIST data according to the file directory path, process by matrix form and save at the appointed file directory in image.

字段 Field

vect_t<feature> elem

数据集元素列

Dataset element set

vect_t<uint64_t> elem_lbl

数据集元素标签列

Dataset element label set

函数 Function

MNIST

构造函数

Constructor

参数 Parameter

声明构造

Construction for declaration

bool bool_preprocess

布尔数值化预处理信号

Boolean digitalization preprocess signal

参数 Parameter

从目标文件目录路径加载指定数量的数据集元素

Load appointed quantity of dataset element from target file directory path

std::string dat_dir

数据集数据路径

Dataset data directory

std::string lbl_dir

数据集标签路径

Dataset label directory

uint64_t qnty

加载数据数量，为 0 时将加载全部数据

Quantity of data loading, all data will be loaded when 0 assigned

uint64_t minibatch

小批次大小

Mini-batch size

bool bool_preprocess

布尔数值化预处理信号

Boolean digitalization preprocess signal

uint64_t padding

矩阵扩展值

Padding value of matrix

参数 Parameter

从目标文件目录路径加载分别指定 MNIST 标签类别数量的数据集元素

Load appointed respectively each label quantity of dataset element from target file directory path

std::string dat_dir

数据集数据路径

Dataset data directory

std::string lbl_dir

数据集标签路径

Dataset label directory

std::initializer_list<uint64_t> qnty_list

各标签加载元素数量表， 若表长度为 1 则每个标签都加载指定数目的元素， 若表长度小于 MNIST 标签类别数值且不为 1，则不进行数据集构造

Quantity list of each label for data loading. Load elements from dataset by the appointed quantity of each label if the size of list is 1 the only appointed value of quantity. Dataset will not construct if the size of list is less than the MNIST label type amount and is not 1.

uint64_t minibatch

小批次大小

Mini-batch size

bool bool_preprocess

布尔数值化预处理信号

Boolean digitalization preprocess signal

uint64_t padding

矩阵扩展值

Padding value of matrix

示例 Example

C++

```
// 加载 MNIST 训练集，为每个标签分别加载 4 个元素
// Load MNIST train set, load 4 elements respectively for every label
MNIST train_set("…\\train-images.idx3-ubyte", "…\\train-labels.idx3-ubyte", {4});
// 加载 MNIST 测试集，分别给每个标签加载 21, 33, 12, 9, 23, 28, 16, 12, 20, 13 个标签
/* Load MNIST test set, load 21, 33, 12, 9, 23, 28, 16, 12, 20, 13 elements
 * respectively for every label
 */
MNIST test_set("…\\t10k-images.idx3-ubyte", "…\\t10k-labels.idx3-ubyte",
                                {21, 33, 12, 9, 23, 28, 16, 12, 20, 13});
```

size

数据集元素数量

Quantity of dataset element

mini_batch

小批次元素数量

Quantity of dataset element in a mini batch

ln_cnt

元素向量行计数

Line count of element vector

col_cnt

元素向量列计数

Line count of element vector

dat_len

单个数据长度

Unit length of data

orgn

获取标签分类矩阵

Get classification matrix of label

返回 Return

所有数据集标签的分类向量集

Classification vector set of all elements label

参数 Parameter

使用 MINST 标签值获取对应向量，静态函数

Get corresponding vector from MNIST label value, static function

uint64_t lbl_val

标签值

Label value

返回 Return

分类向量

Classification vector

load_data

从目标文件目录路径加载数据集元素

Load dataset element from target file directory path

参数 Parameter

从目标文件目录路径加载指定数量的数据集元素

Load appointed quantity of dataset element from target file directory path

std::string dat_dir

数据集数据路径

Dataset data directory

std::string lbl_dir

数据集标签路径

Dataset label directory

uint64_t load_qnty

加载数据数量，为 0 时将加载全部数据

Quantity of data loading, all data will be loaded when 0 assigned

uint64_t minibatch

小批次大小

Mini-batch size

bool bool_preprocess

布尔数值化预处理信号

Boolean digitalization preprocess signal

uint64_t padding

矩阵扩展值

Padding value of matrix

返回 Return

成功加载验证

Successfully load validation

参数 Parameter

从目标文件目录路径加载分别指定 MNIST 标签类别数量的数据集元素

Load appointed respectively each label quantity of dataset element from target file directory path

std::string dat_dir

数据集数据路径

Dataset data directory

std::string lbl_dir

数据集标签路径

Dataset label directory

std::initializer_list<uint64_t> qnty_list

各标签加载元素数量表， 若表长度为 1 则每个标签都加载指定数目的元素， 若表长度小于 MNIST 标签类别数值且不为 1，则不进行数据加载

Quantity list of each label for data loading. Load elements from dataset by the appointed quantity of each label if the size of list is 1 the only appointed value of quantity. Data will not load if the size of list is less than the MNIST label type amount and is not 1.

uint64_t minibatch

小批次大小

Mini-batch size

bool bool_preprocess

布尔数值化预处理信号

Boolean digitalization preprocess signal

uint64_t padding

矩阵扩展值

Padding value of matrix

返回 Return

成功加载验证

Successfully load validation

output_bitmap

输出 MNIST 数据，保存到指定格式的图片

Output MNIST data save as image by appointed format

参数 Parameter

std::string dir_root

目标根目录

Target directory root

uint64_t format

扩展名，格式

Extend, format

[BMIO_PNG][BMIO_JPG][BMIO_GIF][BMIO_TIF][BMIO_BMP]

返回 Return

成功输出验证

Successfully output validation

reset

重置数据集

Reset dataset

neunet

头文件，神经网络库.

Head file, neural network library.

neunet::NetBase

基础网络，定义了一些处理函数范本.

Basic network, define some model of function for process.

函数 Function

NetBase

构造函数

Constructor

参数 Parameter

uint64_t iDscType

梯度下降方式

Gradient descent type

[GD_BGD]　　批量梯度下降 Batch gradient descent

[GD_MINIBATCH_SGD]　Mini-batch 随即梯度 Mini-batch stochastic gradient descent

double dNetAcc

网络训练精度

Network training accuracy

AddLayer

添加网络层，模板函数

Add network layers, model function

参数 Parameter

Args ... pacArgs

用于构造网络层的初始化参数列

Initialization parameters for network layer

返回 Return

添加成功验证

Successfully add validation

Depth

获取网络深度

Get network depth

ShowIter

获取网络深度

Get network depth

ShowIter

迭代监视，模板函数

Iteration monitoring, model function

IterateFlag

迭代信号，用于判定迭代是否继续进行，函数模板

Iteration signal, judge if the iteration continues, model function

返回 Return

迭代继续验证

Iteration continues validation

(bool)

ForwProp

网络前向传播，模板函数

Network forward propagation, model function

返回 Return

前向传播成功验证

Forward propagation successfully validation

(bool)

BackProp

网络反向传播，模板函数

Network back propagation, model function

返回 Return

反向传播成功验证

Forward propagation successfully validation

(bool)

Deduce

推测函数，模板函数

Deduction function, model function

返回 Return

推测结果

Deduction result

(set<vect>)

Run

运行网络，模板函数

Network runs, model function

ValueAssign

数值赋值，将另一个变量的基础数值成员对基础数值成员变量进行赋值，虚函数

Value assignment, assignment the basic type members by another instance, virtual function

参数 Parameter

NetBase netSrc

赋值源

Assignment source

派生 Derivation

neunet::NetClassify

neunet::NetClassify

分类网络，定义了监视，迭代判定等函数

Classification network, define monitoring, iteration function.

函数 Function

NetClassify

构造函数

Constructor

参数 Parameter

uint64_t iDscType

梯度下降方式

Gradient descent type

[GD_BGD]　　批量梯度下降 Batch gradient descent

[GD_MINIBATCH_SGD]　Mini-batch 随即梯度 Mini-batch stochastic gradient descent

double dNetAcc

网络训练精度

Network training accuracy

bool bShowIter

迭代监视信号

Iteration monitoring signal

IterShow

监视前向传播后标签输出

Monitor the output of each label after forward propagation

参数 Parameter

单个批次

A single batch

set<vect> setPreOutput

上一次迭代的输出向量集

Last iteration vector set

set<vect> setCurrOutput

当前迭代的输出向量集

Current iteration vector set

set<vect> setOrigin

标签分类向量

Classification vector of labels

参数 Parameter

多批次

Multi-batch

vect_t<vect> batPreOutput

上一次迭代的输出向量集批次

Last iteration vector set batch

vect_t<vect> batCurrOutput

当前迭代的输出向量集批次

Current iteration vector set batch

vect_t<vect> batOrigin

标签分类向量批次

Classification vector batch of labels

IterFlag

迭代信号，用于判定迭代是否继续进行

Iteration signal, judge if the iteration continues

参数 Parameter

单个批次

A single batch

set<vect> setCurrOutput

当前迭代的输出向量集

Current iteration vector set

set<vect> setOrigin

标签分类向量

Classification vector of labels

参数 Parameter

多批次

Multi-batch

vect_t<vect> batCurrOutput

当前迭代的输出向量集批次

Current iteration vector set batch

vect_t<vect> batOrigin

标签分类向量批次

Classification vector batch of labels

示例 Example

网络的搭建可以分为两种：函数式网络，封装式网络。这两种网络的最大区别就是主要结构的类型，前者是以流程来完成收敛的，后者就是使用模块进行拼接。这个小节将介绍这两种简单的网络搭建方法。

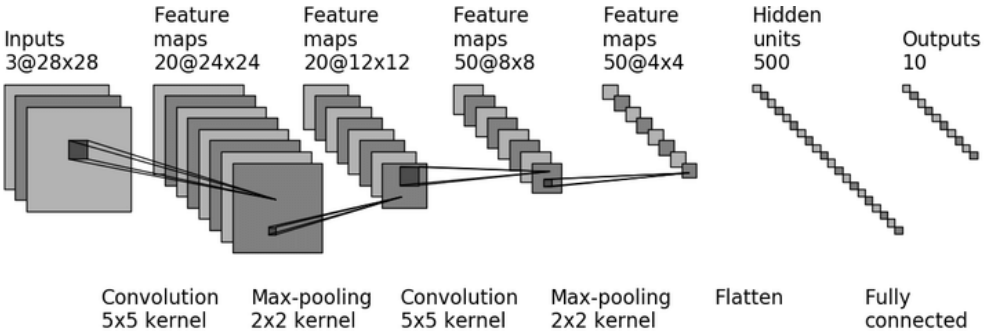
There two types of network construction: Functional network, encapsulated network. The most difference between these two networks is the main structure type, the former completes the convergence by procedure but combined by modules the latter. This chapter will introduce these two simple methods of network construction.

函数式网络 Functional network
封装式网络 Encapsulated network

函数式网络 Functional network

这个例子将使用 LeNet5 进行演示

This sample will use network LeNet5 to introduce



C++

```
// 包含头文件
#include “netbatlib”
#include “dataset”
#include “funclib”
// 对网络需要使用的变量进行声明
// Declare the variables for network using
// c1 层卷积核，AdaDelta 数据，输出
// c1 kernels, AdaDelta data, output
auto kernel_c1 = conv::InitKernel(20, 1, 5, 5);
ada::ada_tensor<ada::AdaDeltaVect> ada_c1(20);
vect beta_c1, gamma_c1;
ada::adaDeltaVect ada_c1_beta, ada_c1_gamma;
set<feature> output_c1;
ConvBN ouput_c1_BN;
set<feature> output_c1_act;
// p2
set<feature> output_p2;
// c3
auto kernel_c3 = conv::InitKernel(50, 20, 5, 5);
ada::ada_tensor<ada::AdaDeltaVect> ada_c3_kernel(50);
vect beta_c3, gamma_c3;
ada::adaDeltaVect ada_c3_beta, ada_c3_gamma;
set<feature> output_c3;
ConvBN ouput_c3_BN;
set<feature> output_c3_act;
// p4
set<feature> output_p4;
// t5
set<vect> output_t5;
// f6
auto weight_f6 = fc::InitWeight(800, 500);
ada::adaDeltaVect ada_f6_weight;
double beta_f6 = 0, gamma_f6 = 1;
ada::AdaDeltaVal ada_f6_beta, ada_f6_gamma
```

```
set<vect> output_f6;
FCBN output_f6_BN;
set<vect> output_f6_act;
// f7
auto weight_f7 = fc::InitWeight(500, 10);
ada::adaDeltaVect ada_f7_weight;
set<vect> output_f7;
set<vect> output_f7_act;
// 定义结束迭代函数
// Define function of the end of iteration
bool iter_valid(set<vect> &output, MNIST &dataset, double acc)
{
    // 如果对应标签值的收敛数值与 1 的差大于精度，那么返回 ture，表示迭代应该继续
    /* If the difference between convergence value of the corresponding label
    *  and 1 is bigger than accuracy, return false, denote the continue of
    *  iteration.
    */
    for(auto i=0; i<output.size(); ++i)
        if(std::abs(output[i][ dataset.elem_lbl[i]][0]-1) > acc)
            return true;
    return false;
}
// main 函数
// Function main
using namespace std;
using namespace dataset;
int main(int argc, char *argv[], char *envp[])
{
    // 载入训练集
    // Load train set
    MNIST dataset("…\\train-images.idx3-ubyte", "…\\train-labels.idx1-ubyte");
    // 获取标签分类向量集
    // Get classification vector set of labels
    auto origin = dataset.orgn();
    // 迭代
    // Iteration
    do for(auto i=0; i<MNIST.elem.size(); ++i)
    {   // 如果未使用 mini-batch
        // If mini-batch not use
        // MNIST.elem.size() = 1
        // 正向传播
        // Propagation
        output_c1 = conv::Conv(MNIST.elem[i], kernel_c1, 1, 1);
        if(!beta_c1.is_matrix()) beta_c1 = conv::BNInitScaleShift(1, 0);
        if(!gamma_c1.is_matrix()) gamma_c1 = conv::BNInitScaleShift(1, 1);
        output_c1_BN = conv::BNTrain(output_c1, beta_c1, gamma_c1);
```

```
        output_c1_act = ReLU(output_c1_BN.setY);

        output_p2 = conv::Pool(output_c1_act, POOL_DOWN_MAX, true,
                                        set<feature>(), 2, 2, 2, 2,);

        output_c3 = conv::Conv(output_p2, kernel_c3, 1, 1);
        if(!beta_c3.is_matrix()) beta_c3 = conv::BNInitScaleShift(1, 0);
        if(!gamma_c3.is_matrix()) gamma_c3 = conv::BNInitScaleShift(1, 1);
        output_c3_BN = conv::BNTrain(output_c3, beta_c3, gamma_c3);
        output_c3_act = ReLU(output_c3_BN.setY);

        output_p4 = conv::Pool(output_c3_act, POOL_DOWN_MAX, true,
                                         blank_ft_seq, 2, 2, 2, 2,);

        output_t5 = fc::FeatureTransform(output_p4);

        output_f6 = fc::Output(output_t5, weight_f6);
        output_f6_BN = fc::BNTrain(output_f6, beta_f6, gamma_f6);
        output_f6_act = sigmoid(output_f6_BN.setY);
        output_f7 = fc::Output(output_f6_act, weight_f7);
        output_f7_act = sigmoid(output_f7);
        // 反向传播
        // Back propagation
        auto grad_output_f7 = softmax_cec_grad(output_f7_act, origin);
        auto grad_f6_act = fc::GradLossToInput(grad_output_f7, weight_f7);
        weight_f7 -= ada_f7_weight.Delta(fc::GradLossToWeight(grad_output_f7,
                                                              output_f6_act));
        auto grad_f6_BN = grad_f6_act.elem_cal_opt(sigmoid_dv(output_f6_BN.setY),
                                                                MATRIX_ELEM_MULT);
        auto grad_f6 = fc::BNGradLossToInput(output_f6_BN.setY, output_f6,
                                                        grad_f6_BN, gamma_f6);
        gamma_f6 -= fc::BNGradLossToScale(output_f6_BN.setY, grad_f6_BN);
        beta_f6 -= fc::BNGradLossToShift(grad_f6_BN);
        auto grad_output_t5 = fc::GradLossToInput(grad_f6_BN, weight_f6);
        weight_f6 -= ada_f6_weight.Delta(fc::GradLossToWeight(grad_f6_BN, output_t5));
        auto grad_output_p4 = fc::FeatureTransform(grad_output_t5,
                        output_p4[0].LN_CNT, output_p4[0].COL_CNT);
        auto grad_output_c3_act = conv::Pool(grad_output_p4, POOL_UP_MAX, false,
                                                        output_c3_act, 2, 2, 2, 2,);
        auto grad_output_c3_BN = grad_output_c3_act.elem_cal_opt(
                    ReLU_dv(output_c3_BN.setY), MATRIX_ELEM_MULT);
        auto grad_output_c3 = conv::BNGradLossToInput(output_c3_BN, output_c3
                                                    grad_output_c3_BN, gamma_c3);
        gamma_c3 = conv::BNAdaDeltaUpdateScaleShift(gamma_c3, conv::BNGradLossToScale(
                                        grad_output_c3_BN,              output_c3_BN),
ada_c3_gamma);
```

```
        beta_c3 = conv::BNAdaDeltaUpdateScaleShift(beta_c3, conv::BNGradLossToShift(
                                                            grad_output_c3_BN),
ada_c3_gamma);
        auto grad_output_p2 = conv::GradLossToInput(grad_output_c3, kernel_c3, 1, 1);
        for(auto i=0; i<ada_c3_kernel.size(); ++i) if(!ada_c3_kernel[i].size())
            ada_c3_kernel[i].init(20);
        kernel_c3 = conv::AdaDeltaUpdateKernel(kernel_c3, conv::GradLossToKernel(
                                grad_output_c3, output_p2, 1, 1), ada_c3_kernel);
        auto grad_output_c1_act = conv::Pool(grad_output_p2, POOL_UP_MAX, false,
                                                            output_c1_act, 2, 2, 2, 2,);
        auto grad_output_c1_BN = grad_output_c1_act.elem_cal_opt(
                      ReLU_dv(output_c1_BN.setY), MATRIX_ELEM_MULT);
        auto grad_output_c1 = conv::BNGradLossToInput(output_c1_BN, output_c1
                                                        grad_output_c1_BN, gamma_c1);
        gamma_c1 = conv::BNAdaDeltaUpdateScaleShift(gamma_c1 conv::BNGradLossToScale(
                                            grad_output_c1BN, output_c1_BN), ada_c1_gamma);
        beta_c1 = conv::BNAdaDeltaUpdateScaleShift(beta_c1, conv::BNGradLossToShift(
                                                            grad_output_c1_BN),
ada_c1_gamma);
        for(auto i=0; i<ada_c1_kernel.size(); ++i) if(!ada_c1_kernel[i].size())
            ada_c1_kernel[i].init();
        kernel_c1 = conv::AdaDeltaUpdateKernel(kernel_c1, conv::GradLossToKernel(
                                grad_output_c1, MNIST.elem[i], 1, 1), ada_c1_kernel);
    }
    while(iter_valid(output_f7_act));
    return EXIT_SUCCESS;
}
```

封装式网络 Encapsulated network

源代码文件 main.cpp 中的 demo 使用的是该方法，具体参见源码.

The demo in source code file main.cpp uses this method, refer to source code for details.