20220526-机器学习

1.学习内容

1.1 机器学习

ANN

Neuron.h   C++   复制代码

```cpp
#pragma once
#ifndef NEURON_H_
#define NEURON_H_

#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <vector>
#define PI 3.141592654
class neuron
{
public:
    neuron(int);
    neuron(double,double,int);
    std::vector<double> weight;
    double generateRandomWeight();
private:
    double miu;
    double sigma;
    int pln;;
};

#endif
```

Neuron.cpp C++ 复制代码

```cpp
#include "neuron.h"

neuron::neuron(int pln_):
    pln(pln_)
{
    if (pln == 0)
    {
        weight.push_back(0);
    }
    else
    {
        miu = 0;
        sigma = 1;
        for (int i = 0; i < pln; i++)
        {
            weight.push_back(generateRandomWeight());
        }
    }
}

neuron::neuron(double miu_,double sigma_,int pln_):
    miu(miu_),sigma(sigma_),pln(pln_)
{
    if (pln == 0)
    {
        weight.push_back(0);
    }
    else
    {
        for (int i = 0; i < pln; i++)
        {
            weight.push_back(generateRandomWeight());
        }
    }
}

double neuron::generateRandomWeight()
{
    double U1 = rand() / double(RAND_MAX);
    double U2 = rand() / double(RAND_MAX);
    double U = std::sqrt(-2 * std::log(U1)) * std::cos(2 * PI * U2);
    double Z = miu + U * sigma;
    return Z;
}
```

Layer.h   C++   复制代码

```cpp
#pragma once
#ifndef LAYER_H_
#define LAYER_H_

#include "neuron.h"
class layer
{
public:
    layer(int,int);
    layer(int, int, std::vector<double>);
    std::vector<neuron*> NeuronList;
private:
    int curLayerNeuronNum;
    int preLayerNeuronNum;
    std::vector<double> rwp;
};

#endif
```

Layer.cpp C++ 复制代码

```cpp
#include "layer.h"

layer::layer(int pln_,int cln_):
    preLayerNeuronNum(pln_),curLayerNeuronNum(cln_)
{
    for (int i = 0; i < curLayerNeuronNum; i++)
    {
        neuron* n = new neuron(preLayerNeuronNum);
        NeuronList.push_back(n);
    }
}

layer::layer(int pln_, int cln_,std::vector<double> rwp_):
    preLayerNeuronNum(cln_), curLayerNeuronNum(pln_),rwp(rwp_)
{
    for (int i = 0; i < curLayerNeuronNum; i++)
    {
        neuron* n = new neuron(rwp[0],rwp[1],preLayerNeuronNum);
        NeuronList.push_back(n);
    }
}
```

Network.h C++ 复制代码

```cpp
#pragma once
#ifndef NETWORK_H_
#define NETWORK_H_

#include "layer.h"
class network
{
public:
    network(int,std::vector<int>);
    network(int, std::vector<int>,std::vector<double>);
    std::vector<layer*> layerList;
private:
    int layerNum;
    std::vector<int> layerNeuronNum;
    std::vector<double> randomWeightParam;
};

#endif
```

Network.cpp C++ 复制代码

```cpp
#include "network.h"

network::network(int ln_,std::vector<int> lnm_)
    :layerNum(ln_),layerNeuronNum(lnm_)
{
    for (int i = 0; i < layerNum; i++)
    {
        if (i == 0)
        {
            layer* lay = new layer(0,layerNeuronNum[i]);
            layerList.push_back(lay);
        }
        else
        {
            layer* lay = new layer(layerNeuronNum[i - 1], layerNeuronNum[i]);
            layerList.push_back(lay);
        }
    }
}

network::network(int ln_, std::vector<int> lnm_, std::vector<double> rwp_)
    :layerNum(ln_), layerNeuronNum(lnm_),randomWeightParam(rwp_)
{
    for (int i = 0; i < layerNum; i++)
    {
        if (i == 0)
        {
            layer* lay = new layer(0, layerNeuronNum[i],randomWeightParam);
            layerList.push_back(lay);
        }
        else
        {
            layer* lay = new layer(layerNeuronNum[i - 1], layerNeuronNum[i],randomWeightParam);
            layerList.push_back(lay);
        }
    }
}
```

2.结果描述

今天基于Neuron–Layer–Network的结构开始设计一个ANN，目前只实现了一小部分，后续还有forward、反向传播以及预测的代码需要实现。争取在回家前完成。