# 20220526-机器学习

```
1.学习内容
1.1 机器学习
ANN
2.结果描述
```

## 1.学习内容

### 1.1 机器学习

#### ANN

```
Neuron.h
                                                                C++ 🗗 🗗 复制代码
 1
     #pragma once
     #ifndef NEURON_H_
 3
     #define NEURON H
 5 ▼ #include <iostream>
     #include <stdlib.h>
 7
     #include <math.h>
     #include <vector>
     #define PI 3.141592654
 9
10
   class neuron
11 ▼ {
12
     public:
          neuron(int);
13
          neuron(double, double, int);
14
          std::vector<double> weight;
15
          double generateRandomWeight();
16
17
     private:
18
          double miu;
19
          double sigma;
20
          int pln;;
21
     };
22
     #endif
23
```

▼ Neuron.cpp C++ □ 复制代码

```
#include "neuron.h"
 1 -
 2
 3
     neuron::neuron(int pln_):
 4
          pln(pln_)
 5 ▼
     {
 6
          if (pln == 0)
 7 🕶
          {
 8
              weight.push_back(0);
 9
          }
          else
10
11 ▼
          {
12
              miu = 0;
              sigma = 1;
13
              for (int i = 0; i < pln; i++)</pre>
14
15 ▼
              {
16
                  weight.push_back(generateRandomWeight());
              }
17
          }
18
19
     }
20
21
      neuron::neuron(double miu_,double sigma_,int pln_):
22
          miu(miu_),sigma(sigma_),pln(pln_)
23 🔻
     {
24
          if (pln == 0)
25 ▼
          {
26
              weight.push_back(0);
27
          }
          else
28
29 🕶
          {
30
              for (int i = 0; i < pln; i++)
31 ▼
              {
32
                  weight.push_back(generateRandomWeight());
33
              }
          }
34
35
     }
36
37
     double neuron::generateRandomWeight()
38 ▼
     {
39
          double U1 = rand() / double(RAND_MAX);
          double U2 = rand() / double(RAND_MAX);
40
          double U = std::sqrt(-2 * std::log(U1)) * std::cos(2 * PI * U2);
41
42
          double Z = miu + U * sigma;
43
          return Z;
     }
44
```

```
Layer.h
                                                              C++ 2 复制代码
     #pragma once
 1
 2
     #ifndef LAYER_H_
     #define LAYER_H_
 3
 4
5 ▼ #include "neuron.h"
     class layer
 6
 7 ▼ {
     public:
 8
9
         layer(int,int);
         layer(int, int, std::vector<double>);
10
         std::vector<neuron*> NeuronList;
11
12
     private:
         int curLayerNeuronNum;
13
         int preLayerNeuronNum;
14
15
         std::vector<double> rwp;
     };
16
17
18
     #endif
```

▼ Layer.cpp C++ C 复制代码

```
1 ▼ #include "layer.h"
 2
     layer::layer(int pln_,int cln_):
 3
          preLayerNeuronNum(pln_), curLayerNeuronNum(cln_)
 4
 5 ▼
     {
          for (int i = 0; i < curLayerNeuronNum; i++)</pre>
 6
 7 🕶
          {
 8
              neuron* n = new neuron(preLayerNeuronNum);
              NeuronList.push_back(n);
 9
          }
10
     }
11
12
     layer::layer(int pln_, int cln_,std::vector<double> rwp_):
13
14
          preLayerNeuronNum(cln_), curLayerNeuronNum(pln_),rwp(rwp_)
15 ▼
     {
          for (int i = 0; i < curLayerNeuronNum; i++)</pre>
16
17 ▼
              neuron * n = new neuron(rwp[0], rwp[1], preLayerNeuronNum);
18
              NeuronList.push_back(n);
19
20
          }
     }
21
```

Network.h C++ 2 复制代码 1 #pragma once 2 #ifndef NETWORK\_H\_ #define NETWORK\_H\_ 4 5 ▼ #include "layer.h" 6 class network 7 ▼ { public: 8 network(int,std::vector<int>); 9 network(int, std::vector<int>,std::vector<double>); 10 std::vector<layer\*> layerList; 11 private: 12 int layerNum; 13 std::vector<int> layerNeuronNum; 14 std::vector<double> randomWeightParam; 15 **}**; 16 17 18 #endif

▼ Network.cpp C++ □ 复制代码

```
#include "network.h"
 1 -
 2
     network::network(int ln_,std::vector<int> lnm_)
 3
 4
          :layerNum(ln_),layerNeuronNum(lnm_)
 5 🔻
     {
          for (int i = 0; i < layerNum; i++)</pre>
 6
 7 -
          {
 8
              if (i == 0)
 9 -
              {
                  layer* lay = new layer(0,layerNeuronNum[i]);
10
                  layerList.push_back(lay);
11
              }
12
13
              else
14 ▼
              {
15
                  layer* lay = new layer(layerNeuronNum[i - 1],
      layerNeuronNum[i]);
16
                  layerList.push_back(lay);
17
              }
          }
18
19
     }
20
21
      network::network(int ln_, std::vector<int> lnm_, std::vector<double>
      rwp_)
          :layerNum(ln_), layerNeuronNum(lnm_), randomWeightParam(rwp_)
22
23 🔻
          for (int i = 0; i < layerNum; i++)</pre>
24
25 ▼
26
              if (i == 0)
27 ▼
28
                  layer* lay = new layer(0,
      layerNeuronNum[i], randomWeightParam);
29
                  layerList.push back(lay);
30
              }
31
              else
32 ▼
              {
                  layer* lay = new layer(layerNeuronNum[i - 1],
33
      layerNeuronNum[i], randomWeightParam);
34
                  layerList.push_back(lay);
35
              }
          }
36
     }
37
```

# 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。