20220607-机器学习

1.学习内容

1.1 机器学习

数据处理

Mnist.h C++ 复制代码

```cpp
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

typedef struct _MnistData
{
    double** pData;
    double** pLabel;
    int nWidth;
    int nHeight;
    int nNumber;
    int nClassNumber;

    _MnistData() :pData(nullptr), pLabel(nullptr), nWidth(0), nHeight(0), nNumber(0), nClassNumber(10) {}
}MnistData, * PMnistData;

//大小节转换
int ReversalInt(int nValue);
//读取Mnist图像数据
bool ReadMnistImage(MnistData& stMnist, const std::string& strPath, int nPadding = 2);
//读取Mnist标签数据
bool ReadMnistLabel(MnistData& stMnist, const std::string& strPath);
//释放Mnist数据
bool ReleaseMnistData(MnistData& stMnist);
```

Mnist.cpp C++ 复制代码

```cpp
#include "Mnist.h"

int ReversalInt(int nValue)
{
    unsigned char cTemp1 = nValue & 255;
    unsigned char cTemp2 = (nValue >> 8) & 255;
    unsigned char cTemp3 = (nValue >> 16) & 255;
    unsigned char cTemp4 = (nValue >> 24) & 255;
    int nData = static_cast<int>(cTemp1) << 24;
    nData += static_cast<int>(cTemp2) << 16;
    nData += static_cast<int>(cTemp3) << 8;
    return nData + cTemp4;
}

bool ReadMnistImage(MnistData& stMnist, const std::string& strPath, int nPadding)
{
    if (strPath.empty()) return false;
    std::fstream MnistFile(strPath, std::fstream::in | std::fstream::binary);
    if (!MnistFile.is_open()) return false;
    int nMagic = 0, nNumber = 0, nWidth = 0, nHeight = 0;
    MnistFile.read(reinterpret_cast<char*>(&nMagic), sizeof(nMagic));
    MnistFile.read(reinterpret_cast<char*>(&nNumber), sizeof(nNumber));
    MnistFile.read(reinterpret_cast<char*>(&nWidth), sizeof(nWidth));
    MnistFile.read(reinterpret_cast<char*>(&nHeight), sizeof(nHeight));

    nMagic = ReversalInt(nMagic);
    if (nMagic != 2051) return false;
    nNumber = ReversalInt(nNumber);
    nWidth = ReversalInt(nWidth);
    nHeight = ReversalInt(nHeight);

    stMnist.nNumber = nNumber;
    stMnist.nWidth = nWidth+nPadding*2;
    stMnist.nHeight = nHeight+nPadding*2;
    double dscaleMax = 1.0;
    double dscaleMin = -1.0;
    int nSize = stMnist.nWidth * stMnist.nHeight;
    stMnist.pData = new double* [stMnist.nNumber];
    for (int i = 0; i < stMnist.nNumber; i++)
    {
        stMnist.pData[i]=new double[nSize];
        for (int j = 0; j < nSize; j++)
        {
```

```
            stMnist.pData[i][j] = -1.0;
        }
        for (int j = 0; j < nHeight; j++)
        {
            for (int k = 0; k < nWidth; k++)
            {
                unsigned char cTemp;
                MnistFile.read(reinterpret_cast<char*>(cTemp),
sizeof(cTemp));
                double dTemp = (static_cast<double>(cTemp) / 255.0) *
(dscaleMax - dscaleMin) + dscaleMin;
                stMnist.pData[i][(j+nPadding)*stMnist.nWidth+k+nPadding]
= dTemp;
            }
        }
    }
    MnistFile.close();
    return true;
}

bool ReadMnistLabel(MnistData& stMnist, const std::string& strPath)
{
    if (strPath.empty()) return false;
    std::fstream MnistLabel(strPath, std::fstream::in |
std::fstream::binary);
    if (MnistLabel.is_open()) return false;
    int nMagic = 0, nNumber = 0;
    MnistLabel.read(reinterpret_cast<char*>(&nMagic), sizeof(nMagic));
    MnistLabel.read(reinterpret_cast<char*>(&nNumber), sizeof(nNumber));
    nMagic = ReversalInt(nMagic);
    nNumber = ReversalInt(nNumber);
    if (nMagic != 2049||nNumber!=stMnist.nNumber) return false;

    char cIndex = 0;
    stMnist.pLabel = new double* [nNumber];
    for (int i = 0; i < nNumber; i++)
    {
        stMnist.pLabel[i] = new double[stMnist.nClassNumber];
        for (int j = 0; j < stMnist.nClassNumber; j++)
        {
            stMnist.pLabel[i][j] == -0.8;
        }
        MnistLabel.read(reinterpret_cast<char*>(&cIndex),
sizeof(cIndex));
        stMnist.pLabel[i][cIndex] = 0.8;
    }
    MnistLabel.close();
    return true;
```

```
}

bool ReleaseMnistData(MnistData& stMnist)
{
    if (stMnist.pData)
    {
        for (int i = 0; i < stMnist.nNumber; i++)
        {
            if (stMnist.pData[i])
            {
                delete[] stMnist.pData[i];
            }
        }
        delete[] stMnist.pData;
        stMnist.pData = nullptr;
    }

    if (stMnist.pLabel)
    {
        for (int i = 0; i < stMnist.nNumber; i++)
        {
            if (stMnist.pLabel[i])
            {
                delete[] stMnist.pLabel[i];
            }
        }
        delete[] stMnist.pLabel;
        stMnist.pLabel = nullptr;
    }

    return true;
}

```

2.结果描述

今天原本预计将Net类也完成，但遇到了一点障碍。明日继续。