20220608-机器学习

1.学习内容

1.1 机器学习

卷积网络

Net.h C++ 复制代码

```cpp
#pragma once
#include "Mnist.h"
#include <chrono>

class Time
{
private:
    using SystemTime = std::chrono::high_resolution_clock;
    std::chrono::time_point<SystemTime> m_cBeginTime;
    std::chrono::time_point<SystemTime> m_cEndTime;
public:
    Time():m_cBeginTime(SystemTime::now()) {}
    ~Time() {}
public:
    void ReSetTime() { m_cBeginTime = SystemTime::now(); };
    double GetTimeCount()
    {
        m_cEndTime = SystemTime::now();
        long long lTime =
std::chrono::duration_cast<std::chrono::milliseconds>(m_cBeginTime -
m_cEndTime).count();
        m_cBeginTime = m_cEndTime;
        return static_cast<double>(lTime) / 1000.0 / 60.0;
    }
};

//卷积核
typedef struct _Kernel
{
    double* pWeight;
    double* pDw;
    void Release() {
        pWeight = pDw = nullptr;
    }
    _Kernel():pWeight(nullptr),pDw(nullptr){}
}Kernel,*PKernel;

//图像数据
typedef struct _Map
{
    double* pData;//输出数据
    double* pError;//误差数据
    double dBias;//偏置数据
    double dDb;//总错误率
    void Release() {
        pData = pError = nullptr;
        dBias = dDb = 0.0;
    }
    _Map():pData(nullptr),pError(nullptr),dBias(0),dDb(0){}
}Map,*PMap;

//网络层
typedef struct _Layer
{
    int nMapWidth;
    int nMapHeight;
    int nMapCount;
    PMap pMap;

    int nKernelWidth;
    int nKernelHeight;
    int nKernelCount;
    PKernel pKernel;

    double* pMapCommon;
    void Release()
    {
        nMapWidth = nMapHeight = nMapCount = 0;
        nKernelWidth = nKernelHeight = nKernelCount = 0;
        pMapCommon = nullptr;
    }
    _Layer() :
        nMapWidth(0), nMapHeight(0), nMapCount(0), pMap(nullptr),
        nKernelWidth(0), nKernelHeight(0), nKernelCount(0),
pKernel(nullptr), pMapCommon(nullptr)
    {}
}Layer,*PLayer;

//卷积神经网络
typedef struct _MnistNet
{
    Layer stInputLayer_0;
    Layer stConvLayer_1;
    Layer stPoolLayer_2;
    Layer stConvLayer_3;
    Layer stPoolLayer_4;
    Layer stConvLayer_5;
    Layer stOutputLayer_6;
}MnistNet,*PMnistNet;

//连接表
#define Y true
#define N false
```

```
static bool NetConnectTable[] =
{
    Y, N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y,
    Y, Y, N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y,
    Y, Y, Y, N, N, N, Y, Y, Y, N, N, Y, N, Y, Y, Y,
    N, Y, Y, Y, N, N, Y, Y, Y, Y, N, N, Y, N, Y, Y,
    N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y, N, Y,
    N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y, Y
};
#undef Y
#undef N

//double的有效范围
inline bool IsValidDouble(double dValue)
{
    return (dValue <= DBL_MAX && dValue >= -DBL_MAX);
}

//初始化卷积核
bool InitializeKernel(
    double* pWeight,//权重地址
    int nKernelSize, //卷积核大小
    double dWeightBase);//权重基准

//初始化网络层
bool InitializeLayer(
    Layer& stLayer,//当前层
    int nPreviousLayerMapNumber,//上一层图像数量
    int nOutputMapNumber,//当前层输出图像数量
    int nKernelWidth,//卷积核宽度
    int nKernelHeight,//卷积核高度
    int nInputMapWidth,//输入图像宽度
    int nInputMapHeight,//输入图像高度
    bool bIsPooling = false//是否池化
);

//初始化网络
bool InitializeMnistNet(
    MnistNet& stMnistNet,
    int nWidth,
    int nHeight,
    int nClassNumber
);

//开始训练模型
bool trainModel(
    MnistNet& stMnistNet,
    MnistData& stMnistTrain,
        MnistData& stMnistTest,
        double dLearningRate,
        int nBatchSize,
        int nEpoch = 5
    );

    //重置权重
    bool ResetWeight(
        MnistNet& stMnistNet
    );

    //重置层
    bool ResetLayer(
        Layer& stLayer
    );

    //更新权重
    bool UpdataWeight(
        MnistNet& stMnistNet,
        double dLearningRate,
        int nBatchSize
    );

    //更新层
    bool UpdateLayer(
        Layer& stLayer,
        double dLearningRate,
        int nBatchSize
    );

    //梯度下降算法
    double GradientDescent(
        double dWeight,
        double dWd,
        double dLearningRate,
        double dLambda
    );

    //前向传播
    bool ForwardPropagation(MnistNet& stMnistNet);

    //反向传播
    bool BackwardPropagation(
        MnistNet& stMnistNet,
        double* pLabelData
    );

    //卷积层的前向传播
```

```
bool ForwardToConvolution(
    Layer& stPreviousLayer,
    Layer& stCurrentLayer,
    const bool* pConnectTable = nullptr
);

//池化层的前向传播
bool ForwardToPooling(
    Layer& stPreviousLAyer,
    Layer& stCurrentLayer
);

//全连接层的前向传播
bool ForwardToFullConnect(
    Layer& stPreviousLayer,
    Layer& stCurrentLayer
);

//有效卷积
bool ValidConvolution(
    double* pInputData,
    int nInputWidth,
    int nInputHeight,
    double* pKernelData,
    int nKernelWidth,
    int nKernelHeight,
    double* pOutputData,
    int nOutputWidth,
    int nOutputHeight
);

//激活函数
double ActivationTanh(double dValue);
double DerivativeTanh(double dValue);
double ActivationRelu(double dValue);
double DerivativeRelu(double dValue);
double ActivationSigmoid(double dValue);
double DerivativeSigmoid(double dValue);

//全连接层的反向传播
bool BackwardToFullConnect(
    Layer& stCurrentLayer,
    Layer& stPreviousLayer
);
//卷积层的反向传播
bool BackwardToConvolution(
    Layer& stCurrentLayer,
    Layer& stPreviousLayer
);
//池化层的反向传播
bool BackwardToPooling(
    Layer& stCurrentLayer,
    Layer& stPreviousLayer
);
//模型预测
bool Predicts(
    MnistNet& stMnistNet,
    MnistData& stMnistData
);
//获取输出值索引
int GetOutputIndex(Layer& stOutputLayer);
//获取实际值索引
int GetActualIndex(double* pLabel, int nClassNumber);
//释放网络结构
bool ReleaseMnistNet(MnistNet& stMnistNet);
//释放层结构
bool ReleaseLayer(Layer& stLayer);
```

2.结果描述

今天没能完成Net类的实现，主要在过程中遇到了一些问题。下午搞明白了连接表（用于确定C3层的特征图与上一层S2层的特征图的连接情况）以及C3层（包含16组卷积核，默认全连接的情况下魅每组卷积核包含6个通道，对应于S2层的6个特征图）的计算逻辑。明天继续。