

# 20220608-机器学习

---

## 1.学习内容

### 1.1 机器学习

#### 卷积网络

## 2.结果描述

## 1.学习内容

### 1.1 机器学习

#### 卷积网络

```
1  #pragma once
2  #include "Mnist.h"
3  #include <chrono>
4
5  class Time
6  {
7  private:
8      using SystemTime = std::chrono::high_resolution_clock;
9      std::chrono::time_point<SystemTime> m_cBeginTime;
10     std::chrono::time_point<SystemTime> m_cEndTime;
11 public:
12     Time():m_cBeginTime(SystemTime::now()) {}
13     ~Time() {}
14 public:
15     void ReSetTime() { m_cBeginTime = SystemTime::now(); };
16     double GetTimeCount()
17     {
18         m_cEndTime = SystemTime::now();
19         long long lTime =
20             std::chrono::duration_cast<std::chrono::milliseconds>(m_cBeginTime -
21                 m_cEndTime).count();
22         m_cBeginTime = m_cEndTime;
23         return static_cast<double>(lTime) / 1000.0 / 60.0;
24     }
25 };
26
27 //卷积核
28 typedef struct _Kernel
29 {
30     double* pWeight;
31     double* pDw;
32     void Release() {
33         pWeight = pDw = nullptr;
34     }
35     _Kernel():pWeight(nullptr),pDw(nullptr){}
36 }Kernel,*PKernel;
37
38 //图像数据
39 typedef struct _Map
40 {
41     double* pData;//输出数据
42     double* pError;//误差数据
43     double dBias;//偏置数据
44     double dDb;//总错误率
45     void Release() {
```

```

44         pData = pError = nullptr;
45         dBias = dDb = 0.0;
46     }
47     _Map():pData(nullptr),pError(nullptr),dBias(0),dDb(0){}
48 }Map,*PMap;
49
50 //网络层
51 typedef struct _Layer
52 {
53     int nMapWidth;
54     int nMapHeight;
55     int nMapCount;
56     PMap pMap;
57
58     int nKernelWidth;
59     int nKernelHeight;
60     int nKernelCount;
61     PKernel pKernel;
62
63     double* pMapCommon;
64     void Release()
65     {
66         nMapWidth = nMapHeight = nMapCount = 0;
67         nKernelWidth = nKernelHeight = nKernelCount = 0;
68         pMapCommon = nullptr;
69     }
70     _Layer() :
71         nMapWidth(0), nMapHeight(0), nMapCount(0), pMap(nullptr),
72         nKernelWidth(0), nKernelHeight(0), nKernelCount(0),
73         pKernel(nullptr), pMapCommon(nullptr)
74     {}
75 }Layer,*PLayer;
76
77 //卷积神经网络
78 typedef struct _MnistNet
79 {
80     Layer stInputLayer_0;
81     Layer stConvLayer_1;
82     Layer stPoolLayer_2;
83     Layer stConvLayer_3;
84     Layer stPoolLayer_4;
85     Layer stConvLayer_5;
86     Layer stOutputLayer_6;
87 }MnistNet,*PMnistNet;
88
89 //连接表
90 #define Y true
91 #define N false

```

```

91     static bool NetConnectTable[] =
92     {
93         Y, N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y,
94         Y, Y, N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y,
95         Y, Y, Y, N, N, N, Y, Y, Y, N, N, Y, N, Y, Y, Y,
96         N, Y, Y, Y, N, N, Y, Y, Y, Y, N, N, Y, N, Y, Y,
97         N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y, N, Y,
98         N, N, N, Y, Y, Y, N, N, Y, Y, Y, Y, N, Y, Y, Y
99     };
100     #undef Y
101     #undef N
102
103     //double的有效范围
104     inline bool IsValidDouble(double dValue)
105     {
106         return (dValue <= DBL_MAX && dValue >= -DBL_MAX);
107     }
108
109     //初始化卷积核
110     bool InitializeKernel(
111         double* pWeight, //权重地址
112         int nKernelSize, //卷积核大小
113         double dWeightBase); //权重基准
114
115     //初始化网络层
116     bool InitializeLayer(
117         Layer& stLayer, //当前层
118         int nPreviousLayerMapNumber, //上一层图像数量
119         int nOutputMapNumber, //当前层输出图像数量
120         int nKernelWidth, //卷积核宽度
121         int nKernelHeight, //卷积核高度
122         int nInputMapWidth, //输入图像宽度
123         int nInputMapHeight, //输入图像高度
124         bool bIsPooling = false //是否池化
125     );
126
127     //初始化网络
128     bool InitializeMnistNet(
129         MnistNet& stMnistNet,
130         int nWidth,
131         int nHeight,
132         int nClassNumber
133     );
134
135     //开始训练模型
136     bool trainModel(
137         MnistNet& stMnistNet,
138         MnistData& stMnistTrain,

```

```

139     MnistData& stMnistTest,
140     double dLearningRate,
141     int nBatchSize,
142     int nEpoch = 5
143 );
144
145 //重置权重
146 bool ResetWeight(
147     MnistNet& stMnistNet
148 );
149
150 //重置层
151 bool ResetLayer(
152     Layer& stLayer
153 );
154
155 //更新权重
156 bool UpdataWeight(
157     MnistNet& stMnistNet,
158     double dLearningRate,
159     int nBatchSize
160 );
161
162 //更新层
163 bool UpdateLayer(
164     Layer& stLayer,
165     double dLearningRate,
166     int nBatchSize
167 );
168
169 //梯度下降算法
170 double GradientDescent(
171     double dWeight,
172     double dWd,
173     double dLearningRate,
174     double dLambda
175 );
176
177 //前向传播
178 bool ForwardPropagation(MnistNet& stMnistNet);
179
180 //反向传播
181 bool BackwardPropagation(
182     MnistNet& stMnistNet,
183     double* pLabelData
184 );
185
186 //卷积层的前向传播

```

```

187     bool ForwardToConvolution(
188         Layer& stPreviousLayer,
189         Layer& stCurrentLayer,
190         const bool* pConnectTable = nullptr
191     );
192
193     //池化层的前向传播
194     bool ForwardToPooling(
195         Layer& stPreviousLayer,
196         Layer& stCurrentLayer
197     );
198
199     //全连接层的前向传播
200     bool ForwardToFullConnect(
201         Layer& stPreviousLayer,
202         Layer& stCurrentLayer
203     );
204
205     //有效卷积
206     bool ValidConvolution(
207         double* pInputData,
208         int nInputWidth,
209         int nInputHeight,
210         double* pKernelData,
211         int nKernelWidth,
212         int nKernelHeight,
213         double* pOutputData,
214         int nOutputWidth,
215         int nOutputHeight
216     );
217
218     //激活函数
219     double ActivationTanh(double dValue);
220     double DerivativeTanh(double dValue);
221     double ActivationRelu(double dValue);
222     double DerivativeRelu(double dValue);
223     double ActivationSigmoid(double dValue);
224     double DerivativeSigmoid(double dValue);
225
226     //全连接层的反向传播
227     bool BackwardToFullConnect(
228         Layer& stCurrentLayer,
229         Layer& stPreviousLayer
230     );
231     //卷积层的反向传播
232     bool BackwardToConvolution(
233         Layer& stCurrentLayer,
234         Layer& stPreviousLayer

```

```

235     );
236     //池化层的反向传播
237     bool BackwardToPooling(
238         Layer& stCurrentLayer,
239         Layer& stPreviousLayer
240     );
241     //模型预测
242     bool Predicts(
243         MnistNet& stMnistNet,
244         MnistData& stMnistData
245     );
246     //获取输出值索引
247     int GetOutputIndex(Layer& stOutputLayer);
248     //获取实际值索引
249     int GetActualIndex(double* pLabel, int nClassNumber);
250     //释放网络结构
251     bool ReleaseMnistNet(MnistNet& stMnistNet);
252     //释放层结构
253     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。