ConvolutionNeuralNetworkHandwriting Recognition

201250068 陈骏

项目内容

通过CNN卷积神经网络实现手写数字识别

环境配置

```
Python 3.7
Package:
    numpy
    cv2
```

项目运行

```
PS D:\MachineLearning\CNNHandwritingRecognition> python .\CNN.py --help
usage: CNN.py [-h] -tf FILEPATH -tl LABELPATH [-cs CONVSIZE] [-cst CONVSTRIDE]
              [-c CHANNELS] [-ps POOLSIZE] [-pst POOLSTRIDE] -t TYPES -TF
              TEST_FILEPATH -TL TEST_LABEL [-time TIME] [-lr LEARNING_RATE]

CNN Handwriting Recognition

optional arguments:
  -h, --help            show this help message and exit
  -tf FILEPATH, --train_file FILEPATH
                        File path of picture
  -tl LABELPATH, --train_label LABELPATH
                        Label file path
  -cs CONVSIZE, --convolution_core_size CONVSIZE
                        Convolution conv core size
  -cst CONVSTRIDE, --convolution_stride CONVSTRIDE
                        Convolution layer stride
  -c CHANNELS, --channels CHANNELS
                        KMeans classify to how many class
  -ps POOLSIZE, --pool_size POOLSIZE
                        Pooling layer size
  -pst POOLSTRIDE, --pool_stride POOLSTRIDE
                        Pooling layer stride
  -t TYPES, --types TYPES
                        Final recognition types
  -TF TEST_FILEPATH, --test_file TEST_FILEPATH
                        Test file path of picture
  -TL TEST_LABEL, --test_label TEST_LABEL
                        Test label path of picture
  -time TIME
  -lr LEARNING_RATE, --learning-rate LEARNING_RATE
```

代码结构

```
D:.
|   ActivatingFunction.py
|   CNN.py  # main function
|   ConvolutionLayer.py
|   FullConnectLayer.py
|   log.txt
|   PicRead.py
|   PoolingLayer.py
|   README.md
|
├─.idea
|   |   .gitignore
|   |   CNNHandwritingRecognition.iml
|   |   misc.xml
|   |   modules.xml
|   |   workspace.xml
|   |
|   └─inspectionProfiles
|           profiles_settings.xml
|           Project_Default.xml
|
├─Resources
|   |   t10k-images.idx3-ubyte
|   |   t10k-labels.idx1-ubyte
|   |   train-images.idx3-ubyte
|   |   train-labels.idx1-ubyte
|   |
|   └─PNGs
```

CNN实现

由卷积层，池化层，全连接层实现卷积神经网络，其中全连接层和池化层为非线性传播，全连接层采用全连接实现线性传播。反向传播中，全连接层采用梯度下降的优化策略，池化层采用最大池化，反向传播仅进行局部传播，卷积层优化采用将每一个误差反馈到卷积核的卷积参数上。详见代码及代码注释。

```
class Convolution:
    filter_size = None
    filter_num = None  # 卷积核的数目是用来确定提取多少个特征，比如一个卷积核用来提取嘴，另一个用来提取手
    filter = None
    bias = None
    padding_size = None
    img = None
    img_shape = None
    padding = None
    strider = 1

    def __init__(self, filter_size, strider=1, filter_num=1):
        """
        卷积层初始化
        :param filter_size: 卷积核大小
        :param strider: 卷积步长
        :param filter_num: 卷积核的数目 / 通道数
        """
```

```
        self.filter_size = filter_size
        self.padding_size = filter_size // 2
        self.stride = strider
        self.filter_num = filter_num
        self.filter = np.random.rand(filter_num, filter_size, filter_size)
        print("Info : convolution layer init")
        print("Info : filter size " + str(filter_size))
        print("Info : filter stride " + str(strider))
        print("Info : filter num / channel " + str(filter_num))

    def conv(self, img):
        """
        进行卷积
        :param img: 传入图片 28 * 28 * 1
        :return: 卷积结果
        """
        print("Info : begin convolution")
        height, width = img.shape
        pixel = np.asarray(img)
        self.img_shape = pixel.shape
        self.img = pixel
        pixel = np.pad(pixel, ((self.padding_size, self.padding_size),
(self.padding_size, self.padding_size)),
                       mode='constant', constant_values=(0, 0))
        self.padding = pixel
        result = []
        # 进行卷积
        # 将一个个视野与卷积核进行直接相乘
        for i in range(0, (height + 2 * self.padding_size - self.filter_size) //
self.stride + 1, self.stride):
            result.append([])
            for j in range(0, (width + 2 * self.padding_size - self.filter_size)
// self.stride + 1, self.stride):
                result[i // self.stride].append(
                        np.sum(self.filter * (pixel[i: i + self.filter_size,
j: j + self.filter_size]), axis=(1, 2)))
        # print("Info : img after convolution " + str(result))
        return result

    def feedback(self, feedback_info, learning_rate):
        """
        卷积层反馈
        :param feedback_info: 池化层反馈结果
        :param learning_rate: 学习率
        """
        print("Info : begin conv feedback")
        # print("Info : feedback " + str(feedback_info))
        # feedback.shape 在 stride 为 1 的情况下为 28 * 28 * 3
        # 与卷积结果相同
        filters = np.zeros(self.filter.shape)
        for i in range(0, (self.img_shape[0] + 2 * self.padding_size -
self.filter_size) // self.stride + 1, self.stride):
            for j in range(0, (self.img_shape[1] + 2 * self.padding_size -
self.filter_size) // self.stride + 1, self.stride):
                for k in range(self.filter_num):
                    # 卷积层反馈，将卷积结果的每一个误差反馈到卷积核的卷积参数上
                    filters[k] += feedback_info[i, j, k] * self.padding[i: i +
self.filter_size, j: j + self.filter_size]
```

```
        self.filter -= learning_rate * filters
```

```
class Pool:
    size = 2
    stride = 2
    input_img = None
    input_img_shape =None

    def __init__(self, stride=2, size=2):
        """
        池化层初始化
        :param stride: 池化步长
        :param size: 池化大小
        """
        self.stride = stride
        self.size = size  # 通常与步长相同
        print("Info : pooling layer init")
        print("Info : pooling size " + str(self.size) + " * " + str(self.size))
        print("Info : pooling stride " + str(self.stride) + " usually the same
as pooling size")

    def max_pooling(self, conv_img):
        """
        采用最大池化
        :param conv_img: 卷积结果
        :return: 池化结果
        """
        print("Info : begin max pooling for conv img")
        height, width, depth = np.asarray(conv_img).shape
        # 最大池化
        conv_img = np.asarray(conv_img)
        self.input_img = conv_img
        self.input_img_shape = self.input_img.shape
        result = []
        for i in range(0, height, self.stride):
            result.append([])
            for j in range(0, width, self.stride):
                result[i // self.stride].append([])
                for k in range(depth):
                    sub = conv_img[i: i + self.size, j: j + self.size, k: k + 1]
                    result[i // self.stride][j //
self.stride].append(np.max(sub))
        # print("Info : img pooled " + str(result))
        # 到这边是14 * 14 * 3 的一个矩阵
        return result

    def feedback(self, feedback_info):
        """
        池化层反馈
        :param feedback_info: 全连接层反馈结果
        :return: 池化层反馈结果
        """
        print("Info : begin pooling layer feedback")
        # print(feedback_info)
        # print(np.asarray(feedback_info).shape)
        # input_nodes = 28 * 28 * 3
        input_nodes = np.zeros(self.input_img_shape)
```

```
        height, width, depth = self.input_img_shape
        for i in range(0, height, self.stride):
            for j in range(0, width, self.stride):
                for k in range(depth):
                    sub = self.input_img[i: i + self.size, j: j + self.size, k:
k + 1]
                    a = np.max(sub)
                    if self.input_img[i][j][k] == a:
                        input_nodes[i][j][k] = feedback_info[i // self.stride][j
// self.stride][k]
        # 将14 * 14 * 3 的池化层的误差回退到卷积层层面
        return input_nodes
```

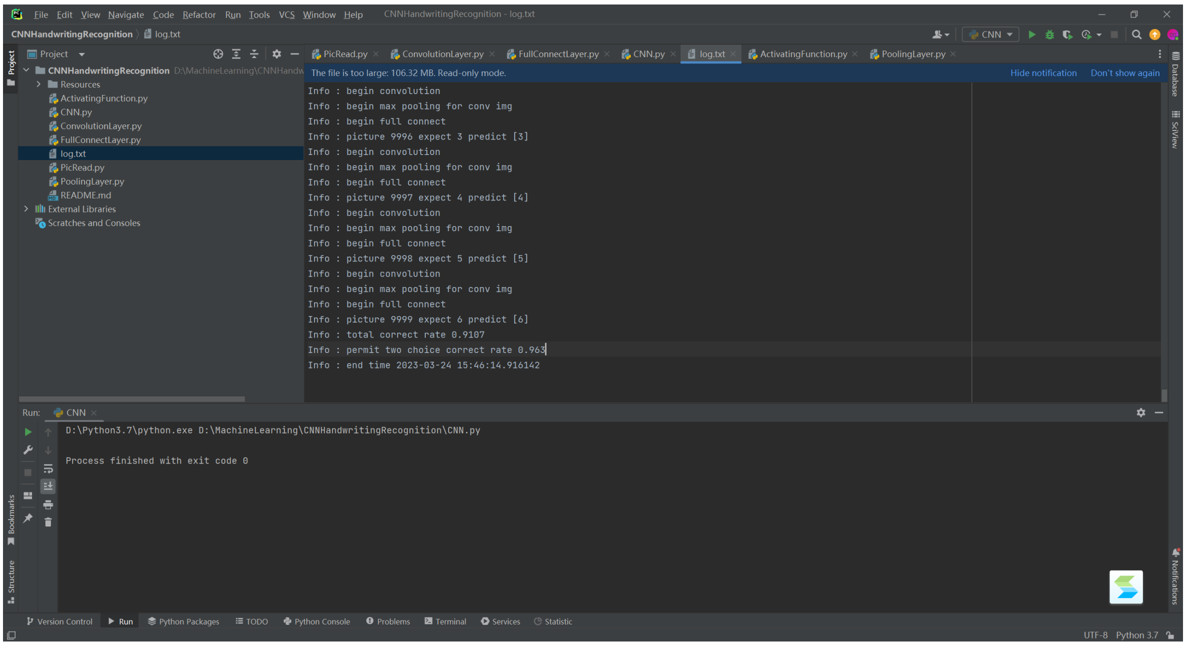
```
class FullConnect:
    full_connect_matrix = None
    offset = None
    gap_matrix = None
    types = None
    input_img = None
    input_img_shape = None
    out = None
    def __init__(self, pic_size, types):
        """
        全连接层初始化
        :param pic_size: 传入的池化后的矩阵大小
        :param types: 所有的类别数目
        """
        self.full_connect_matrix = np.random.rand(pic_size, types) / pic_size
        self.offset = np.zeros(types)
        print(self.full_connect_matrix)
        print("Info : full connect layer init")
        print("Info : full connect matrix size " +
str(self.full_connect_matrix.shape))
        print("Info : full connect offset size " + str(self.offset.shape))
        pass
    """
    作全连接，根据前面的Convolution和Pooling层计算之后，28*28的灰度矩阵变成了14*14*3的矩
阵
    Height * Width 变成了 (Height / (Conv.stride * Pool.stride))  * (Width /
(Conv.stride * Pool.stride)) * Conv.filter_num
    """
    def full_connect(self, pool_img):
        """
        进行全连接
        :param pool_img: 池化结果
        :return: 分类预测，为一个一维的，大小为类别数的矩阵
        """
        print("Info : begin full connect")
        img_array = np.asarray(pool_img)
        self.input_img = img_array.flatten()
        self.input_img_shape = img_array.shape
        height, width, depth = img_array.shape

        img_flatten = img_array.flatten()
        result = np.dot(img_flatten, self.full_connect_matrix) + self.offset
        self.last_total = result
```

```
        out = np.exp(result)
        # print("Info : full connect result " + str(result))
        return out / np.sum(out, axis=0)

    def full_connect_feedback(self, gradients, learn_rate):
        """
        全连接层反向传播
        :param gradients: 预测差
        :param learn_rate: 学习率
        :return:
        """
        print("Info : begin full connect feedback")
        # print("Info : gradients " + str(gradients))
        for i, gradient in enumerate(gradients):  # i为下标，gradient为具体的值
            # print(gradients)
            if gradient != 0:
                exps = np.exp(self.last_total)
                s = np.sum(exps)

                out_back = -exps[i] * exps / (s ** 2)
                # 反馈
                out_back[i] = exps[i] * (s - exps[i]) / (s ** 2)
                # 将反馈数值和概率做乘积，得到结果权重1
                out_back = gradient * out_back

                # 最后的输出与结果反馈的权重做点乘，获得权重的偏置
                weight_back = self.input_img[np.newaxis].T @
out_back[np.newaxis]
                inputs_back = np.dot(self.full_connect_matrix, out_back)
                # print("Info : full connect matrix before " +
str(self.full_connect_matrix))
                print("Info : predict before " + str(np.dot(self.input_img,
self.full_connect_matrix) + self.offset))
                self.full_connect_matrix -= learn_rate * weight_back
                # print("Info : full connect matrix after " +
str(self.full_connect_matrix))
                print("Info : predict after " + str(np.dot(self.input_img,
self.full_connect_matrix) + self.offset))
                self.offset -= learn_rate * out_back

                # 将矩阵从 1d 转为 3d
                # 588 to 14x14x3
        return inputs_back.reshape(self.input_img_shape)
```

```
beta = 0.005
def Sigmoid(x):
    # 激活函数，将线性关系转化为非线性关系
    g_x = 1 / (1 + np.exp(-beta * x))
    return g_x
```

训练集与测试集

采用http://yann.lecun.com/exdb/mnist/数据集，由idx3-ubyte文件加载60000份训练集及10000份测试集。训练结果指标采用完全正确率及前二正确率，即预测完全正确以及正确结果在预测前二可能的情况中的比例。测试结果如下


File Edit View Navigate Code Refactor Run Tools VCS Window Help CNNHandwritingRecognition - log.txt
CNNHandwritingRecognition log.txt
Project
CNNHandwritingRecognition
Resources
ActivatingFunction.py
CNN.py
ConvolutionLayer.py
FullConnectLayer.py
log.txt
PicRead.py
PoolingLayer.py
README.md
External Libraries
Scratches and Consoles
PicRead.py ConvolutionLayer.py FullConnectLayer.py CNN.py log.txt ActivatingFunction.py PoolingLayer.py
The file is too large: 106.32 MB. Read-only mode.
Hide notification Don't show again
Info : begin convolution
Info : begin max pooling for conv img
Info : begin full connect
Info : picture 9996 expect 3 predict [3]
Info : begin convolution
Info : begin max pooling for conv img
Info : begin full connect
Info : picture 9997 expect 4 predict [4]
Info : begin convolution
Info : begin max pooling for conv img
Info : begin full connect
Info : picture 9998 expect 5 predict [5]
Info : begin convolution
Info : begin max pooling for conv img
Info : begin full connect
Info : picture 9999 expect 6 predict [6]
Info : total correct rate 0.9107
Info : permit two choice correct rate 0.963
Info : end time 2023-03-24 15:46:14.916142
Run: CNN
D:\Python3.7\python.exe D:\MachineLearning\CNNHandwritingRecognition\CNN.py
Process finished with exit code 0
Version Control Run Python Packages TODO Python Console Problems Terminal Services Statistic
UTF-8 Python 3.7