人工智能导论实验报告

姓　　名：陈博

学　　号：U202214123

班　　级：人工智能 2204 班

任课教师：郑定富、昌毅

成　　绩：

时　　间：2024 年春季学期

单　　位：人工智能与自动化学院

目录

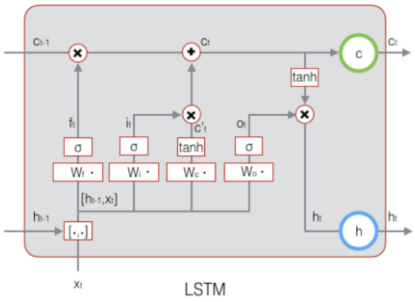
目录

1、实验 1　循环神经网络 NLP-情感分类

实验内容

本次实验是情感分析问题，指判断一段文本所表达的情绪状态，属于文本分类问题。

本次实验主要通过循环神经网络 LSTM 进行对序列的预测，具体的思路如下图所示：



LSTM

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

每个单元包含以下几个门：

1. 遗忘门：用来控制记忆消失程度。
2. 输入门：决定了当前时刻的输入信息，有多少信息将添加到记忆信息流中，与遗忘门计算公式几乎一致，输入门同样通过一个激活函数来实现。
3. 记忆状态：计算当前输入与过去的记忆所具有的信息总量。
4. 输出门：控制着有多少记忆信息将被用于下一阶段的更新中。

然后我们构建神经网络：

```
# 定义长短期记忆网络

def lstm_net(ipt, input_dim):

    # 以数据的IDs作为输入

    emb = fluid.layers.embedding(input=ipt, size=[input_dim, 128], is_sparse=True)

    # 第一个全连接层

    fc1 = fluid.layers.fc(input=emb, size=128)

    # 进行一个长短期记忆操作

    lstm1, _ = fluid.layers.dynamic_lstm(input=fc1, #返回：隐藏状态（hidden state），LSTM的神经元状态

                                         size=128) #size=4*hidden_size

    # 第一个最大序列池操作

    fc2 = fluid.layers.sequence_pool(input=fc1, pool_type='max')

    # 第二个最大序列池操作

    lstm2 = fluid.layers.sequence_pool(input=lstm1, pool_type='max')

    # 以softmax作为全连接的输出层，大小为2,也就是正负面

    out = fluid.layers.fc(input=[fc2, lstm2], size=2, act='softmax')

    return out
```

将损失函数设置为交叉熵损失函数，训练含有 50%的正面评价和 50%的负面评价的训练集，用 Adagrad 算法，最后进行测试。

实验结果

预测的结果如下图所示：

```
运行时长: 23毫秒  结束时间: 2024-07-11 14:40:00

'read the book forget the movie'的预测结果为：正面概率为：0.43615，负面概率为：0.56385
'this is a great movie'的预测结果为：正面概率为：0.35888，负面概率为：0.64112
'this is very bad'的预测结果为：正面概率为：0.37659，负面概率为：0.62341
```

实验分析

尽管我们成功地训练了一个 LSTM 模型来进行情感分析，但测试集的准确率较低，但通过增加训练轮数和调整超参数，可以进一步提高准确率，优化模型性能。

2、实验 2　深度学习入门 NLP-文本分类

实验内容

我们从网站上爬取 56821 条数据中文新闻摘要，包含 10 种类别，国际、文化、娱乐、体育、财经、汽车、教育、科技、房产、证券。我们使用卷积神经网络对它进行分类

我们定义了卷积神经网络（CNN）结构，包括嵌入层、卷积层、池化层和 Softmax 层，如下图所示：

```
def CNN_net(data,dict_dim, class_dim=10, emb_dim=128, hid_dim=128,hid_dim2=9
    emb = fluid.layers.embedding(input=data,
                                 size=[dict_dim, emb_dim])
    conv_3 = fluid.nets.sequence_conv_pool(
                                           input=emb,
                                           num_filters=hid_dim,
                                           filter_size=3,
                                           act="tanh",
                                           pool_type="sqrt")
    conv_4 = fluid.nets.sequence_conv_pool(
                                           input=emb,
                                           num_filters=hid_dim2,
                                           filter_size=4,
                                           act="tanh",
                                           pool_type="sqrt")

    output = fluid.layers.fc(
        input=[conv_3, conv_4], size=class_dim, act='softmax')
    return output
```

同时，我们定义交叉熵损失函数和准确率函数，设置 Adagrad 优化算法进行训练，训练过程如下图所示：

```
Pass:4, Batch:0, Cost:0.64107, Acc:0.79688
Pass:4, Batch:100, Cost:0.70728, Acc:0.77344
Pass:4, Batch:200, Cost:0.58543, Acc:0.83594
Pass:4, Batch:300, Cost:0.59938, Acc:0.85156
Test:4, Cost:0.71917, ACC:0.76265
Pass:5, Batch:0, Cost:0.52632, Acc:0.85156
Pass:5, Batch:100, Cost:0.53485, Acc:0.82812
Pass:5, Batch:200, Cost:0.60272, Acc:0.78906
Pass:5, Batch:300, Cost:0.62984, Acc:0.83594
Test:5, Cost:0.71243, ACC:0.76535
Pass:6, Batch:0, Cost:0.51439, Acc:0.83594
Pass:6, Batch:100, Cost:0.65486, Acc:0.76562
Pass:6, Batch:200, Cost:0.55180, Acc:0.82031
Pass:6, Batch:300, Cost:0.61714, Acc:0.81250
Test:6, Cost:0.71315, ACC:0.76708
Pass:7, Batch:0, Cost:0.60437, Acc:0.79688
Pass:7, Batch:100, Cost:0.57546, Acc:0.82031
Pass:7, Batch:200, Cost:0.50731, Acc:0.85938
Pass:7, Batch:300, Cost:0.70080, Acc:0.72656
Test:7, Cost:0.71244, ACC:0.76647
Pass:8, Batch:0, Cost:0.50716, Acc:0.84375
Pass:8, Batch:100, Cost:0.48080, Acc:0.84375
Pass:8, Batch:200, Cost:0.55230, Acc:0.81250
Pass:8, Batch:300, Cost:0.73101, Acc:0.75781
Test:8, Cost:0.70865, ACC:0.76569
Pass:9, Batch:0, Cost:0.53858, Acc:0.82812
Pass:9, Batch:100, Cost:0.49069, Acc:0.85156
Pass:9, Batch:200, Cost:0.42450, Acc:0.85156
Pass:9, Batch:300, Cost:0.66774, Acc:0.78125
Test:9, Cost:0.71265, ACC:0.76682
训练模型保存完成!
```

实验结果

最后我们的测试结果如下图所示：

```
预测结果标签为：0，  名称为：文化，  概率为：0.886722
预测结果标签为：8，  名称为：国际，  概率为：0.431292
```

与实际相比：

```
data1 = get_data('在获得诺贝尔文学奖7年之后，莫言15日晚间在山西汾阳贾家庄如是说')
data2 = get_data('综合“今日美国”、《世界日报》等当地媒体报道，芝加哥河滨警察局表示，')
```

可见结果符合我们的预期

实验分析

通过简单几层的 CNN，就可以实现很好的分类效果，并能够给出相应的概率，如果有更多轮的训练，或者有更多且适当的层数，效果应该会更好。

3、实验 3　卷积神经网络实践-猫狗分类

实验内容

本次实验解决图像分类的问题，图像分类是根据图像的语义信息将不同类别图像区分开来，是计算机视觉中重要的基本问题，猫狗分类属于图像分类中的粗粒度分类问题，我们使用卷积神经网络来解决这个问题。

我们使用 CIFAR10 数据集。CIFAR10 数据集包含 60,000 张 32x32 的彩色图片，10 个类别，每个类包含 6,000 张。其中 50,000 张图片作为训练集，10000 张作为验证集。这次我们只对其中的猫和狗两类进行预测。

我们构建如下所示的卷积神经网络：

```
def convolutional_neural_network(img):
    # 第一个卷积-池化层
    conv_pool_1 = fluid.nets.simple_img_conv_pool(
        input=img,          # 输入图像
        filter_size=5,      # 滤波器的大小
        num_filters=20,     # filter 的数量。它与输出的通道相同
        pool_size=2,        # 池化核大小2*2
        pool_stride=2,      # 池化步长
        act="relu")         # 激活类型
    conv_pool_1 = fluid.layers.batch_norm(conv_pool_1)
    # 第二个卷积-池化层
    conv_pool_2 = fluid.nets.simple_img_conv_pool(
        input=conv_pool_1,
        filter_size=5,
        num_filters=50,
        pool_size=2,
        pool_stride=2,
        act="relu")
    conv_pool_2 = fluid.layers.batch_norm(conv_pool_2)
    # 第三个卷积-池化层
    conv_pool_3 = fluid.nets.simple_img_conv_pool(
        input=conv_pool_2,
        filter_size=5,
        num_filters=50,
        pool_size=2,
        pool_stride=2,
        act="relu")
    # 以softmax为激活函数的全连接输出层，10类数据输出10个数字
    prediction = fluid.layers.fc(input=conv_pool_3, size=10, act='softmax')
    return prediction
```

训练时，我们使用交叉熵损失函数来衡量分类任务的误差，并定义准确率函数来评估模型性能，并使用 Adam 优化方法，设置学习率为 0.001。

Executor:接收传入的 program，通过 run()方法运行 program。

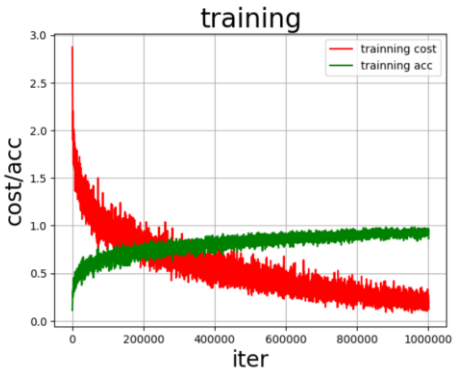
训练时，首先定义运算场所，使用 CPU 或 GPU 进行训练。创建 Executor 并初始化参数。使用 DataFeeder 将数据转化为特定结构，供 Executor 使用。

Executor 接收传入的 program,并根据 feed map(输入映射表)和 fetch_list(结果获取表) 向 program 中添加 feed operators(数据输入算子)和 fetch operators（结果获取算子)。 feed map 为该 program 提供输入数据。fetch_list 提供

program 训练结束后用户预期的变量。每一个 Pass 训练结束之后，再使用验证集进行验证，并打印出相应的损失值 cost 和准确率 acc。

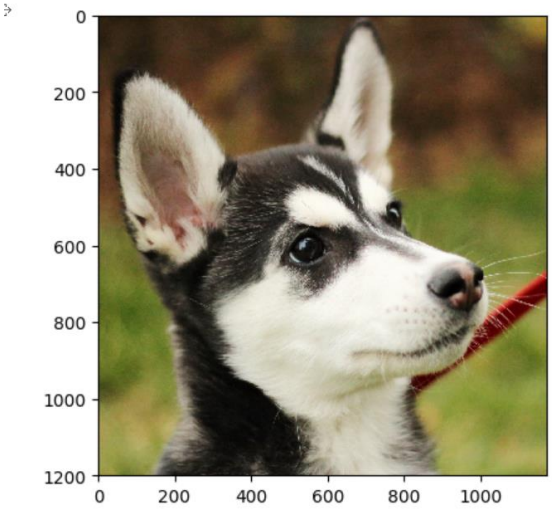
训练结果如下图所示：

```
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/matplotlib/cbook/__init__.py:2349: DeprecationWarning: Using or importing the ABCs fr
nstead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  if isinstance(obj, collections.Iterator):
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/matplotlib/cbook/__init__.py:2366: DeprecationWarning: Using or importing the ABCs fr
nstead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  return list(data) if isinstance(data, collections.MappingView) else data
```



实验结果

测试结果如下图所示：




```
im_shape的维度:  (1, 3, 32, 32)
results [array([[3.8811977e-05, 2.6314663e-02, 3.1785294e-07, 1.2032729e-01,
        1.9718131e-08, 3.4844413e-01, 1.0822589e-04, 3.6026626e-07,
        5.0476617e-01, 1.2460589e-09]], dtype=float32)]
infer results: ship
```

实验分析

本次实验构建了卷积神经网络来学习图像分类问题，但最后的结果分类错误，可能是由于过拟合的原因，因为 training 的准确率相当高。

4、实验 4　波士顿房价预测

实验内容

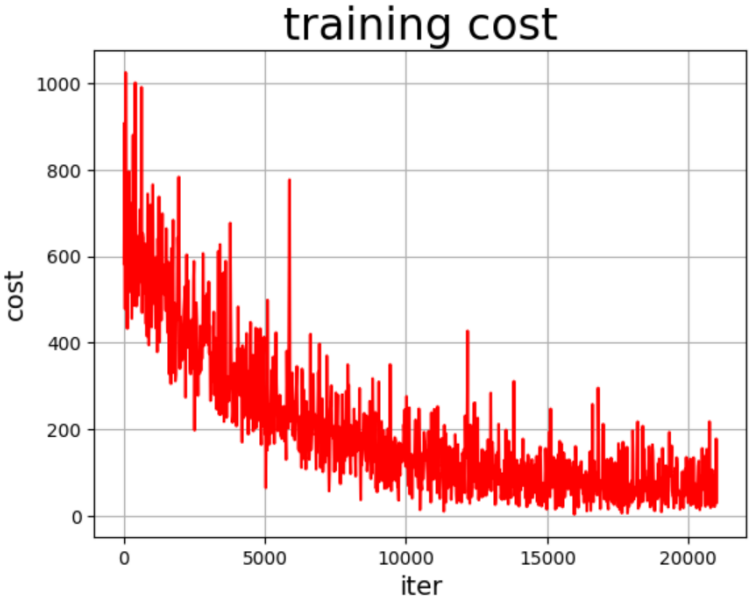
经典的线性回归模型主要用来预测一些存在着线性关系的数据集。回归模型可以理解为：存在一个点集，用一条曲线去拟合它分布的过程。如果拟合曲线是一条直线，则称为线性回归。如果是一条二次曲线，则被称为二次回归。线性回归是回归模型中最简单的一种。 本次实验使用 PaddlePaddle 建立起一个房价预测模型。

我们使用 UCI 房价数据集，该数据集包含 506 行，每行 14 列。前 13 列描述房屋的各种信息，最后一列为房屋价格中位数。使用 PaddlePaddle 提供的接口读取 uci_housing 训练集和测试集。通过 paddle.reader.shuffle()进行数据随机打乱，通过 paddle.batch()将数据分批次读取。

通过建立一个简单的线性网络，即一个从输入到输出的全连接层。输入为 13 维的特征值，输出为预测的房价，使用均方差损失函数来衡量预测值与真实值之间的误差、随机梯度下降（SGD）优化算法来调整模型参数。

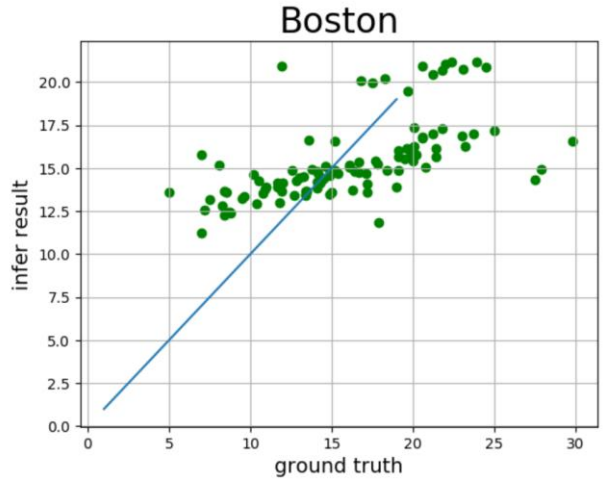
训练过程如下图所示：

```
Save models to /home/aistudio/work/fit_a_line.inference.model
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/matplotlib/cbook/__init__.py:2349: D
nstead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  if isinstance(obj, collections.Iterator):
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/matplotlib/cbook/__init__.py:2366: D
nstead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  return list(data) if isinstance(data, collections.MappingView) else data
```



实验结果

```
97: infer:21.20   gt:22.40
98: infer:20.94   gt:20.60
99: infer:21.17   gt:23.90
100: infer:21.08   gt:22.00
101: infer:20.93   gt:11.90
```



实验结果如上图所示。

实验分析

可见，线性模型预测的值和真实值还是有一定的差距，但效果已经很好了。因此，房价的估计可以近似看作线性回归模型，但是为了更精确的估计，还是需要非线性单元。

5、体会与建议

通过这几次的实验，我认识到了神经网络的重要性，以及他所能完成的任务的多样性，从情感分析到图像分类，同时，我也认识到模型复杂性与训练程度、参数的关键作用。

对于本课程实验的建议，实验内容我觉得很有意思，但我希望能够在学期中就把任务发布下来。