机器学习实验 – 2023春

实验五：深度学习

20202060287 李昂

实验内容：深度学习

环境要求：

Python，numpy支持多维度的数组和矩阵运算，pandas数据处理和分析工具，Matplotlib图形化工具，pytorch深度学习库，tensorflow深度学习库，中文分词工具jieba

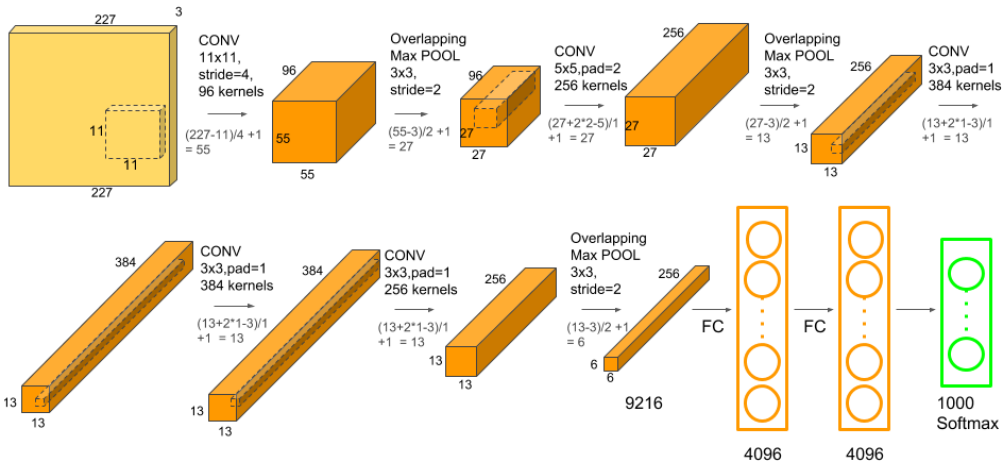
```
import torch
import torch.optim as optim
from torch import nn
from torch.utils.data import DataLoader
from torch.utils.tensorboard import SummaryWriter
from torchvision import transforms
from torchvision.datasets import CIFAR10
--------------------------------------------------------------------------------
from os import path
from re import sub, compile

import numpy as np
import tensorflow as tf
from gensim.models import Word2Vec
from jieba import cut
from keras import callbacks
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.losses import BinaryCrossentropy  # 二分类交叉熵损失函数
from keras.models import Sequential  # 顺序模型
from pandas import read_csv, DataFrame
from sklearn.model_selection import train_test_split  # 划分训练集和测试集
```

1　任务一、卷积神经网络

本任务中你将使用深度学习库pytorch完成图像分类任务，搭建卷积神经网络的经典模型AlexNet，完成对cifar-10数据的分类任务，记录训练过程中的损失和准确率以及测试集的损失和准确率，并将其可视化，分析结果。

文件夹cifar-10-batches-py包含我们的图像分类问题的数据集,data_batch_1,2..5为训练数据，test_batch为测试数据,为了方便图片转成张量，使用torchvision加载和处理数据集。图像加载预处理示例代码见ex5-1.py。

模型细化结构如下图所示



1.1 分析

> 该任务通过搭建卷机神经网络CNN完成图像分类任务。这里直接创建AlexNet，继承`torch.nn`,根据上图CNN结构，构造神经网络，使用`torch`提供的`optim`模块优化，最后使用`tensorboard`记录训练日志，并保存模型。

该任务的难点在于构建AlexNet，对于初学者，仅根据上图直接搭建有些吃力，这里参考了：

1. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
2. https://discuss.pytorch.org/t/given-input-size-256x1x1-calculated-output-size-256x0x0/40941

这两篇Pytorch官方的教程和社区答疑比较细致的讲解了如何通过torch搭建一般的CNN以及AlexNet的细节，作为本次实验的参考。

另外由于使用的是m1 MAC，torch无法调用GPU完成训练（本人调试了很久，效果极不理想）因此这里使用CPU完成，消耗了不少时间。

1.2 代码实现

```
# -*- coding: utf-8 -*-
# @Time : 5/12/23 16:19
# @Author : ANG

import torch
import torch.optim as optim
from torch import nn
from torch.utils.data import DataLoader
from torch.utils.tensorboard import SummaryWriter
from torchvision import transforms
from torchvision.datasets import CIFAR10

# device = "cuda:0" if torch.cuda.is_available() else "cpu"
```

```
device = "cpu"
# 设定超参数
BATCH_SIZE = 64  # 设定每一个Batch的大小
EPOCHS = 15  # 迭代次数


# 定义AlexNet模型
class AlexNet(nn.Module):
    def __init__(self) -> None:
        super(AlexNet, self).__init__()
        # Cov2d参数分别为：输入通道数，输出通道数，卷积核大小，步长，填充
        self.conv1 = nn.Sequential(nn.Conv2d(3, 96, 11, 4, 2), nn.ReLU(),
nn.MaxPool2d(3, 2))
        self.conv2 = nn.Sequential(nn.Conv2d(96, 256, 5, 1, 2), nn.ReLU(),
nn.MaxPool2d(3, 2))
        self.conv3 = nn.Sequential(nn.Conv2d(256, 384, 3, 1, 1), nn.ReLU())
        self.conv4 = nn.Sequential(nn.Conv2d(384, 384, 3, 1, 1), nn.ReLU())
        self.conv5 = nn.Sequential(nn.Conv2d(384, 256, 3, 1, 1), nn.ReLU(),
nn.MaxPool2d(3, 2))
        self.fc1 = nn.Sequential(nn.Linear(256 * 6 * 6, 4096), nn.ReLU(),
nn.Dropout(0.5))
        self.fc2 = nn.Sequential(nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.5))
        self.fc3 = nn.Linear(4096, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = self.conv5(x)
        x = x.view(x.shape[0], -1)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x


if __name__ == "__main__":
    # 数据预处理
    dataPath = "/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/
数据集"  # 数据集路径，填写'cifar-10-batches-py'文件夹的上级路径
    transform = transforms.Compose([transforms.ToTensor(), transforms.Resize(224),
transforms.RandomHorizontalFlip(),
                                    transforms.Normalize(mean=[0.5], std=[0.5])])
    trainData = CIFAR10(dataPath, train=True, transform=transform, download=False)  #
下载训练集和测试集
    testData = CIFAR10(dataPath, train=False, transform=transform, download=False)
```

```
    trainDataLoader = DataLoader(dataset=trainData, batch_size=BATCH_SIZE,
shuffle=True)  # 构建数据集和测试集的DataLoader
    testDataLoader = DataLoader(dataset=testData, batch_size=BATCH_SIZE,
shuffle=False)
    classes = ("Airplane", "Car", "Bird", "Cat", "Deer", "Dog", "Frog", "Horse",
"Ship", "Truck")

    # AlexNet
    # 实例化网络、优化器、损失函数
    net = AlexNet().to(device)  # 实例化网络
    optimizer = optim.Adam(net.parameters())  # 实例化优化器
    criterion = nn.CrossEntropyLoss()  # 实例化损失函数
    # 训练模型，并使用tensorboard可视化
    writer = SummaryWriter(log_dir="/Users/wallanceleon/Desktop/机器学习/机器学习实
验/20201060287-李昂-实验五/日志及训练模型/AlexNet_Logs")
    for epoch in range(EPOCHS):
        for i, data in enumerate(trainDataLoader, 0):
            inputs, labels = data
            # inputs = inputs.cuda()
            # labels = labels.cuda()
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            writer.add_scalar("loss", loss.item(), epoch * len(trainDataLoader) + i)
    # 保存模型
    torch.save(net.state_dict(), "/Users/wallanceleon/Desktop/机器学习/机器学习实
验/20201060287-李昂-实验五/日志及训练模型/AlexNet模型/AlexNet.pth")
```

1.3 实验结果

2 任务二、循环神经网络

本任务中你将深度学习工具tensorflow.keras完成中文文本的情感分析，搭建循环神经网络中的长短期记忆网络LSTM模型，使用keras提供的文本嵌入分词处理工具，完成对中文评论数据集的情感分析。将数据集留出30%作为测试集，记录训练过程的损失和准确率以及测试集的损失和准确率，并将其可视化，分析结果。

文件comments.csv包含我们的中文评论的数据集，comments代表一条评论，label代表文本情感（好评/差评）。以及文本预处理的中文停等词文件chineseStopWord.txt。文本预处理示例代码见ex5-2.py。

2.1 分析

- 数据预处理：读取数据，利用jieba将comments列拆分为词列表。
- 使用当前语料训练Word2Vec模型，并利用该模型将上述词列表转换为索引矩阵。
- 定义LSTM模型，划分测试集和训练集。
- 预先一个tensorboard的实例回调，然后在训练模型时将其添加到callback参数中，让tensorboard获取到数据源，以实现可视化。

2.2 代码实现

```
# -*- coding: utf-8 -*-
# @Time : 5/12/23 16:42
# @Author : ANG

from os import path
from re import sub, compile

import numpy as np
import tensorflow as tf
from gensim.models import Word2Vec
from jieba import cut
from keras import callbacks
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.losses import BinaryCrossentropy  # 二分类交叉熵损失函数
from keras.models import Sequential  # 顺序模型
from pandas import read_csv, DataFrame
from sklearn.model_selection import train_test_split  # 划分训练集和测试集

MAX_NB_WORDS = 50000  # 设置最频繁使用的50000个词
MAX_SEQUENCE_LENGTH = 250  # 每条cut_review最大的长度
EMBEDDING_DIM = 100  # 设置Embedding层的维度
# 设定超参数
BATCH_SIZE = 100  # 批处理大小
EPOCHS = 30  # 迭代次数
LSTM_UNITS = 64  # LSTM单元数


def dataprocess(data: DataFrame, stopwords: list):
    """
    数据预处理，将data['comments']直接分成词列表
    :param data:
    :return:
    """
```

```
    content1 = data.replace(" ", "")  # 去掉文本中的空格
    pattern = compile("[^\u4e00-\u9fa5^a-z^A-Z^0-9]")  # 只保留中英文、数字，去掉符号
    content2 = sub(pattern, "", str(content1))  # 把文本中匹配到的字符替换成空字符
    return [w for w in list(cut(content2)) if w not in stopwords]  # 直接分词，
data['comments']格式为list[list[str]]


def stopwordlist(filepath):
    """
    加载停用词表
    :param filepath:
    :return:
    """
    stopwords = [line.strip() for line in open(filepath, "r", encoding="utf-
8").readlines()]
    return stopwords


def trainWord2Vec(data, savePath: str = None) -> Word2Vec:
    """
    训练Word2Vec模型并保存
    :param savePath: Word2Vec模型的路径
    :return: Word2Vec模型
    """
    model = Word2Vec(data["comments"].values, vector_size=EMBEDDING_DIM, window=5,
workers=8)
    if not savePath:
        savePath = "w2v.model"
    model.save(savePath)  # 保存模型
    return model


def data2Index(data, savePath: str = None) -> np.ndarray:
    """
    将data转换为索引矩阵
    :param data:
    :return:
    """
    indexMatrix = np.zeros((len(data), MAX_SEQUENCE_LENGTH), dtype="int32")
    for i in range(len(data)):
        review = data["comments"][i]  # 一条评论
        for j in range(len(review)):
            if j < MAX_SEQUENCE_LENGTH:  # 限制每条评论的长度，只取前250个词
                word = review[j]
                if word in wordsList:
                    indexMatrix[i][j] = wordsList.index(word)
                else:
                    indexMatrix[i][j] = 0  # 未出现在词表中的词用0表示
```

```
    # 保存data的索引矩阵
    if not savePath:
        savePath = "indexMatrix.npy"
    np.save(savePath, indexMatrix)
    return indexMatrix


if __name__ == "__main__":
    # 数据预处理
    stopwords = stopwordlist("/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/数据集/chineseStopWords.txt")  # 读取停用词表
    data = read_csv("/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/数据集/comments.csv")  # 读取数据集
    data["comments"] = data.comments.apply(dataprocess, args=(stopwords,))  # 将data['comments']处理为list[str]的分词形式
    print(data.info)  # 查看数据集信息
    print(data.head())  # 查看数据集前5行

    # 训练Word2Vec模型，将词转换为词向量
    w2vModel = (Word2Vec.load("w2v.model") if path.exists("w2v.model") else trainWord2Vec(data, "w2v.model"))
    wordsList = w2vModel.wv.index_to_key  # 获取词表
    wordVectors = w2vModel.wv.vectors  # 获取词向量矩阵
    print("wordVectors.shape:", wordVectors.shape)  # Word2Vec模型的词向量矩阵

    # 将data转换为索引矩阵
    indexMatrix = (
        np.load("indexMatrix.npy") if path.exists("indexMatrix.npy") else data2Index(data, "indexMatrix.npy"))
    print("indexMatrix.shape:", indexMatrix.shape)  # 索引矩阵的尺度应该约为(9918)

    # 划分训练集和测试集
    X_train, X_test, y_train, y_test = train_test_split(indexMatrix, data["label"], test_size=0.3)
    X_train = X_train.squeeze()

    # LSTM
    # 搭建LSTM模型
    model = Sequential()
    model.add(Embedding(input_dim=MAX_NB_WORDS, output_dim=EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH))
    model.add(LSTM(activation="sigmoid", units=LSTM_UNITS, return_sequences=True))
    model.add(Dropout(0.5))
    model.add(Dense(units=512, activation="relu"))
    model.add(Dropout(0.5))
    model.add(Dense(units=1, activation="sigmoid"))
    model.summary()
    model.compile(loss=BinaryCrossentropy(), optimizer="adam", metrics=["accuracy"])
```

```
    # 训练模型，并使用tensorboard可视化
    writer = tf.summary.create_file_writer("/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/日志及训练模型/LSTM_Logs")
    tb_callback = callbacks.TensorBoard(log_dir="/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/日志及训练模型/LSTM_Logs", histogram_freq=1)
    model.fit(X_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, callbacks=[tb_callback])
    # 保存模型
    model.save("/Users/wallanceleon/Desktop/机器学习/机器学习实验/20201060287-李昂-实验五/日志及训练模型/LSTM模型/LSTM.h5")
    # 评估模型
    score = model.evaluate(X_test, y_test)
    print("Test loss:", score[0])
    print("Test accuracy:", score[1])
```

2.3 实验结果

经过30轮迭代，在训练集上的正确率为98.05%；在测试集上的损失为0.3413，准确率为87.98%。