实验2：影评情感分类器

1. 题目

题目：请基于电影评论数据，训练一个评论情感的二分类器，要求：训练电影评论数据的词向量，对于每个评论，使用其**词向量的平均值**来表示，并通过**逻辑回归**模型判断测试集的评论是正面（1）还是负面（0），输出测试集预测精度。

2. 代码

```
import torch
import torch.nn as nn
import torch.optim as optim
import jieba
import numpy as np
from sklearn.preprocessing import StandardScaler
from gensim.models import Word2Vec

# 示例数据：电影评论与标签（1 表示正面，0 表示负面）
sentences = [
    "这部电影太精彩了，演员表现很棒！",  # 1（正面）
    "太难看了，浪费了我两个小时。",  # 0（负面）
    "音乐很好听，但剧情太俗套了。",  # 0（负面）
    "看完之后好开心，完全值得推荐！",  # 1（正面）
    "一般般，没什么特别的地方。",  # 0（负面）
]
labels = [1, 0, 0, 1, 0]  # 目标分类标签

# 1. 数据预处理：分词
tokenized_corpus = [list(jieba.cut(sentence)) for sentence in sentences]

# 2. 使用 gensim 训练 Word2Vec 模型
word2vec_model = Word2Vec(sentences=tokenized_corpus, vector_size=100, window=5,
min_count=1, workers=4)

# 3. 创建每个句子的向量表示：计算句子中所有词向量的平均值
def sentence_to_vector(sentence, model):
    words = list(jieba.cut(sentence))
    word_vectors = [model.wv[word] for word in words if word in model.wv]
    if len(word_vectors) > 0:
        return np.mean(word_vectors, axis=0)
    else:
        return np.zeros(model.vector_size)

X = np.array([sentence_to_vector(sentence, word2vec_model) for sentence in
sentences])

# 4. 标准化特征（可选）
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 5. 转换为 PyTorch Tensor
X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
```

```
y_tensor = torch.tensor(np.array(labels, dtype=np.float32).reshape(-1, 1),
dtype=torch.float32)

# 6. 逻辑回归模型
class LogisticRegressionModel(nn.Module):
    def __init__(self, input_size):
        super().__init__()
        self.linear = nn.Linear(input_size, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        return self.sigmoid(self.linear(x))

# 7. 训练模型
model = LogisticRegressionModel(X.shape[1])
criterion = nn.BCELoss()  # 二元交叉熵损失
optimizer = optim.Adam(model.parameters(), lr=0.01)

for epoch in range(200):
    optimizer.zero_grad()
    predictions = model(X_tensor)
    loss = criterion(predictions, y_tensor)
    loss.backward()
    optimizer.step()

    if epoch % 50 == 0:
        print(f'Epoch {epoch}, Loss: {loss.item():.4f}')

# 8. 预测新的句子
new_sentences = ["这部电影好感人，剧情很精彩！", "太烂了，毫无逻辑。"]
new_X = np.array([sentence_to_vector(sentence, word2vec_model) for sentence in
new_sentences])
new_X_scaled = scaler.transform(new_X)
new_X_tensor = torch.tensor(new_X_scaled, dtype=torch.float32)

predicted_probs = model(new_X_tensor).detach().numpy()
predicted_labels = (predicted_probs > 0.5).astype(int)

print("Predicted Sentiments:", predicted_labels)
```