# 作家风格识别

**姓名**：胡天扬

**学号**：3190105708

**专业**：自动化（控制）

**课程**：人工智能与机器学习

**指导教师**：张建明 徐巍华

## 题目分析

本题考察 `NLP` 部分的内容，需要 `RNN` 的知识做铺垫，并结合 `预训练` 的操作，才能获得较高的评分。

## 神经网络

原题给的网络模型是 `LSTM+FL`，但是由于没有预训练，相当于随机进行词向量的编码，因此正确率只有 `五分之一` 左右，考虑更改网络模型：

```
model = nn.Sequential(
    nn.Linear(word_num, 1024),
    nn.ReLU(),
    nn.Linear(1024, 1024),
    nn.ReLU(),
    nn.Linear(1024, 5))
```

将超参数和损失函数等设置为如下：

| batch size | epoch | lr | loss | optimizer |
|---|---|---|---|---|
| 32 | 20 | 1e-4 | CrossEntropyLoss | Adam |

由于没有限定模型初始化时的权重，所以每次训练出的模型都有略微差别，在本地测试集上的准确率大约在0.94，而在提交测试结果上预测的成功率有45/50，也就是 0.9 左右。

## GloVe

用预训练好的 `word2vec` 和 `GloVe` 进行尝试，如 `glove.6B.300d.txt` ，并且在神经网络中加入 `nn.embedding` 层，模型如下：

```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.embedding = nn.Embedding(len(TEXT.vocab), 300)
        self.lstm = nn.LSTM(input_size=300, hidden_size=512, num_layers=4)
        self.fc1 = nn.Linear(512, 1024)
        self.fc2 = nn.Linear(1024, 5)
```

但是效果不佳，正确率反而有所下降，只有0.7左右，可能是参数没调好。

## BERT

考虑可以用更“高级”的预训练模型，尝试了 `ELMo` 和 `BERT` 两种，据说 `ELMo` 对中文词库更为友好，但是本题试了下感觉 `BERT` 效果更好。模型如下：

```
model = BertForSequenceClassification.from_pretrained('bert-base-chinese', num_labels=num_labels, output_attentions=False, output_hidden_states=False)
```

并且加入预热阶段，使用 `transformers.get_linear_schedule_with_warmup`，让学习率从0线性增加到优化器中的初始 `lr`，然后再线性下降到 `0`。

```
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)
```

另外，对于长度为 `T` 的序列，我们在迭代中计算这T个时间步上的梯度， 将会在反向传播过程中产生长度为 `O(T)` 的矩阵乘法链。当 `T` 较大时，可能导致数值不稳定， 例如梯度爆炸或梯度消失，所以要进行梯度裁剪：

```
torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
```

训练后的准确率果然有明显提升，在提交测试中准确率有 49/50，唯一的缺点就是训练时间太长，用 `GPU` 都需要二十分钟。

## 心得

`word2vec` 和 `GloVe` 都将相同的预训练向量分配给同一个词，而不考虑词的上下文。而随着 **上下文敏感** 词表示的发展，`TagLM`、`CoVe`、`ELMo` 等模型不断出现。其中，`ELMo` 为输入序列中的每个单词分配一个表示的函数，这显著改进了各种自然语言处理任务的解决方案，但每个解决方案仍然依赖于一个特定于任务的架构。之后又出现了**任务无关** 的 `GPT`，在 $Transformer$ 解码器的基础上，预训练了一个用于表示文本序列的语言模型。于是为了结合 `ELMo` 的上下文编码和 `GPT` 的从左至右编码这两个优势， `BERT` 诞生了。通过使用预训练的 $Transformer$ 编码器， `BERT` 能够基于其双向上下文表示任何词元。而在下游任务的监督学习过程中， `BERT` 又与 `GPT` 相似。在本题中的诸多方法中，果然还是 `BERT` 展现了很大的优势。