Final 1

171098160 王慧敏 商学院金融工程专业

一、 请分别采用 RNN（LSTM），Bert（Bert family）架构模型进行训练模型，并预测，说明你的 RNN/Bert 的模型架构、训练过程 (learning curve) 和准确率为何？

采用 labeled train data 进行训练
（一）LSTM 模型

1、预处理：

- tokenize：对 labeled train data 和 test data 中的每一条评论用 jieba 分词转化为词列表
- 词语筛选：取 labeled train data 和 test data 的所有评论，统计所有评论中每个词各出现了多少次，只保留出现次数高于 min（实验设置为 5）的词；将各条评论中未被保留的词，用<unk>替代
- Padding：将所有评论统一成 sen_len（实验设置为 20）长，词数大于 sen_len 的句子直接截断，小于 sen_len 的句子用<pad>补齐
- 构建 Word2idx 字典: 对所有词（包括<pad>、 <unk>）进行编号，一个词对应一个 id
- 将评论的词列表转化为 index 列表：通过查阅 Word2idx 字典，将每条评论的词列表转化为相应的 index 的列表
- 构建 Embedding：用 troch.nn.Embedding 进行随机 Embedding，embedding_dim=250

2、模型结构

```
# LSTM层
self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=num_layers, batch_first=True)
# output全连接层
self.classifier = nn.Sequential( nn.Dropout(dropout),
                                 nn.Linear(hidden_dim, 64),
                                 nn.Dropout(dropout),
                                 nn.Linear(64,1),
                                 nn.Sigmoid() )
```

实验中取 LSTM 的 hidden_dim=150，num_layers=1
分类器采用带一个隐藏层的全连接网络，设置 dropout=0.3

3、训练

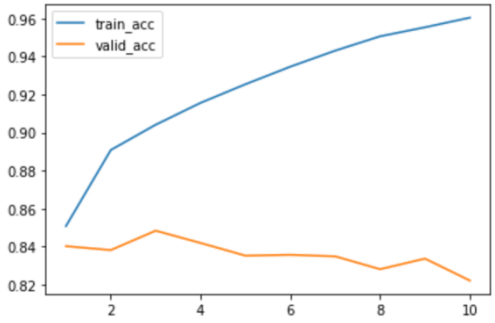
把 training data 取 90%为 train（18 万个），10%为 valid（2 万个）

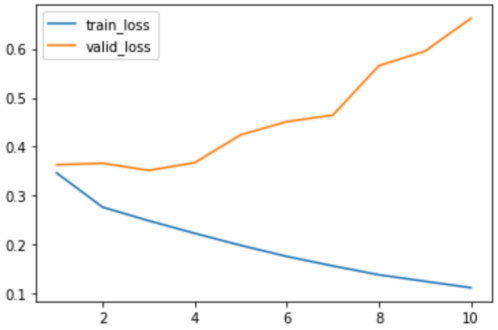
Optimizer 采用 Adam，设置 learning rate = 0.001

设置 batch_size = 128，训练 10 个 epoch

4、结果

- 训练集正确率 Train acc 一直升高，在训练 10 个 epoch 后达到 96.029%
- 验证集正确率 Val acc 先上升后波动下降，在第 3 个 epoch 最高，达到 84.838%





（二）Bert 模型

1、预处理

- tokenize：采用 Bert pretrain Tokenizer，设置 tokenizer = BertTokenizer.from_pretrained('bert-base-chinese')，利用 tokenizer.encode_plus，对 labeled train data 和 test data 中的每一条评论进行 tokenize、限制最长长度、并直接转化为 ids 形式，得到 input_ids、token_type_ids
- Padding，将所有评论统一成 sen_len（实验设置为 20）长，词数大于 sen_len 的句子直接截断，小于 sen_len 的句子用[PAD]的 ids 补齐

2、模型

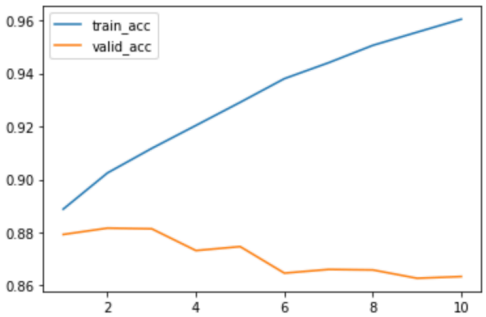
直接采用 BertForSequenceClassification 的 pre_train 模型：
model = BertForSequenceClassification.from_pretrained("bert-base-chinese", config=config)
config 中设置 output class 有两类

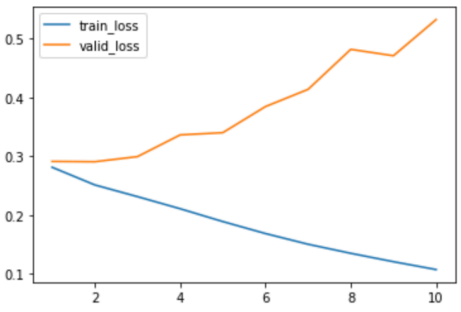
3、训练

把 training data 取 90%为 train，10%为 valid
设置 batch_size = 128，训练 10 个 epoch
Optimizer 使用 AdamW，learning rate = 1e-5，weight_decay = 1e-2

4、结果

- 训练集正确率 Train acc 一直升高，最终 10 个 epoch 后达到 96.045%
- 验证集正确率 Val acc 初始就很高，第 1 个 epoch 就达到 87.928%，在第 2 个 epoch 达到峰值 88.167%，随后波动下降，最终 10 个 epoch 后得到 86.341%

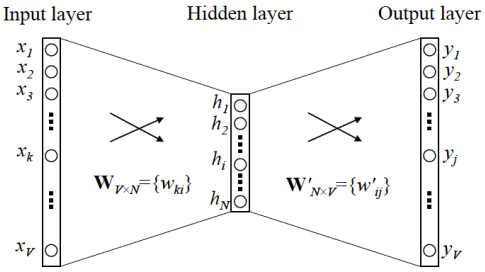


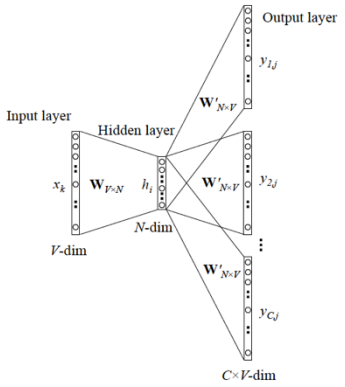


二、请叙述你如何 improve performance（preprocess、embedding、架构等等），并解释为何这些做法可使模型进步，并列出准确率与 improve 前的差异

在第一问中的 LSTM 模型上进行修改，主要改变预处理中的 Embedding 的方法，其他预处理、网络结构、训练方法均不改变

1、预处理：
用 Word2Vec 的方法改进第一问中的随机 Embedding。Word2Vec 在生成 Embedding 的时候能够考虑词语间的上下文关系。Word2Vec 有两种形式：Ski-gram（用当前词来预测上下文）和 CBOW（通过上下文来预测当前值）





以 Skip-gram 模型为例，简化网络结构如上左图。词典中总共有 V 个词，输入是 input word 的 V 维 one-hot encoder，经过一个无激活函数的隐藏层，输出是词典中每个词有多大可能性跟 input word 同时出现。Embedding 即为 Hidden layer 的输出。当考虑上下文多个词语的时候，网络结构如上右图，可以看成是多个简单模型的并联，cost function 是单个模型 cost function 的累加（取 log 之后）

在训练中，采用 labeled train data 和 test data 中的所有评论训练 Word2Vec 模型。采用 gensim.models 的 word2vec 包，选择 embedding 维度 size=250，词向量上下文最大距离（滑动窗口）window=5，保留的最低词频 min_count=5，用于控制训练的并行数 workers=12，设置随机梯度下降法中迭代的最大次数 iter=10，选择 Skip-Gram 模型 sg=1

```
x = train_x + test_x
word2vec_model = word2vec.Word2Vec(x, size=250, window=5,
min_count=5, workers=12, iter=10, sg=1)
```

2、网络结构
与第一问中的 LSTM 模型相同

```
# LSTM层
self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=num_layers, batch_first=True)
# output全连接层
self.classifier = nn.Sequential( nn.Dropout(dropout),
                                 nn.Linear(hidden_dim, 64),
                                 nn.Dropout(dropout),
                                 nn.Linear(64,1),
                                 nn.Sigmoid() )
```

3、训练

与第一问中的 LSTM 模型相同：

把 training data 取 90%为 train，10%为 valid

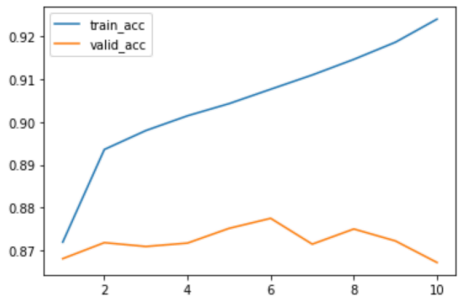
Optimizer 采用 Adam，设置 learning rate = 0.001

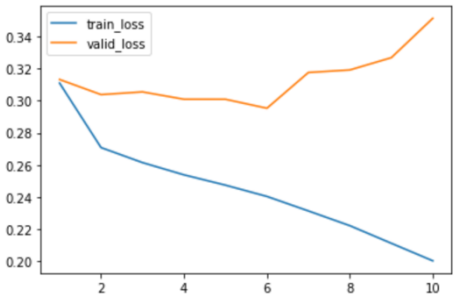
设置 batch_size = 128，训练 10 个 epoch = 10

4、结果

- 训练集正确率 Train acc 一直升高，在第 10 个 epoch 达到最高，但是 10 个 epoch 后得到的 train acc 比第一问中的 LSTM 模型要低，train acc = 92.398%（vs 第一问中的 LSTM 模型为 96.029%）
- 验证集正确率 Val acc 先上升后波动下降，整体比第一问中的 LSTM 模型水平高，且达到峰值需要训练的 epoch 数更多，第 6 个 epoch 达到最高（vs 第一问中的 LSTM 模型在第 3 个 epoch 中就达到峰值），val acc = 87.754%（vs 第一问中的 LSTM 模型峰值为 84.838%）

相比第一问中的 LSTM 模型，采用 Word2Vec Embedding 后，训练集准确率更低、验证集准确率更高，训练集和验证集的准确率差值更低，说明 Word2Vec 能够降低对 train data 的过拟合，提高训练准确性。推测是由于 Word2Vec Embedding 考虑了语义，Embedding 带来的随机性较第一问的 Embedding 更小，作为网络的 Input，使得模型在训练时更少的去拟合随机性，因此能够降低过拟合，同时会带来更好的结果





三、（选做）请描述你的 semi-supervised 方法是如何标记 label，并比较有无 semi-supervised training 对准确率的影响并试着探讨原因

Self-training 方法：

1、 预处理

对 train_x、unlabeled_x、text_x 进行分词、padding、word2ids，得到 embedding，模仿上一问， Embedding 采用 Word2Vec 模型。与上一问的差别在于，Word2Vec 模型是采用 labeled train data、unlabeled train data 和 test data 中的所有评论训练得到的

Word2Vec 训练方法与上一问相同，采用 gensim.models 的 word2vec 包，选择 embedding 维度 size=250，词向量上下文最大距离（滑动窗口）window=5，保留的最低词频 min_count=5，用于控制训练的并行数 workers=12，设置随机梯度下降法中迭代的最大次数 iter=10，选择 Skip-Gram 模型 sg=1

```
x = train_x + unlabeled_x + test_x
word2vec_model = word2vec.Word2Vec(x, size=250, window=5, min_count=5, workers=12, iter=10, sg=1)
```

2、 取 labeled data 中的 90%为训练集，10%为验证集进行训练。网络模型、optimizer、learning rate、batch size 均一致，同样训练 10 个 epoch

3、 用得到的模型预测 unlabeled train data，得到预测可信度高的 unlabeled data，并标注相应的 label 加入到训练集中。预测可信度的判断通过 threshold 来判定，设定 threshold=0.9，将 input 得到的 prediction > 0.9 或 < 0.1 的 data 视为预测可信度高的数据，其中将得到的 prediction > 0.9 的 input 设置为 label = 1、prediction < 0.1 的 input 设置为 label = 0。共得到置信度高的 unlabeled data 73 万 3062 个

4、 将置信度高的 unlabeled train data 加入到训练集中（18 万个 labeled data、以及 73 万 3062 个置信度高的 unlabeled train data 共同组成训练集），2 万个原有的 labeled data 为验证集

5、 重新训练模型，网络结构、optimizer、learning rate、batch size、epoch 均与上一问中的 LSTM 模型相同

结果：

- 训练集正确率 Train acc 一直保持在很高的水平，且一直在升高，从第一个 epoch 的 97.043%上升到第 10 个 epoch 的 98.659%，远远高于之前模型的 train acc
- 验证集正确率 Val acc 整体保持稳定，在第 4 个 epoch 达到最高值 88.027%，比不采用 self training 的 LSTM 模型的 val acc 略有提升（不采用 self training 的最高值为 87.893%），但是改善效果不是特别明显

推测是由于 self training 训练集中 unlabeled data 是根据相同模型在只用 labeled data 训练后筛选得到的，这些置信度高的 unlabeled data 数量非常多，达到 73 万，导致大量拥有相同 label 的有很大的共性（因为是通过之前训练得到的 label），所以自然会很容易导致 train acc 非常高；而训练得到的模型与不使用 self training 时不会有太大差别，导致 val acc 不会有很大改善