# 基于深度学习的情感分类

1. 模型评价指标（好听的话还要自己扩充）

在完成对分类模型的训练之后，需要对训练好的模型进行性能评价，即比较分类模型的预测值与数据真实值之间的误差。在分类器训练集上的误差为训练误差，在测试集上的误差为泛化误差。我们希望训练一个泛化误差尽可能小的分类器。泛化误差的评估是比较难的，一般是分类模型在测试集合上的近似。

对模型的泛化性能评估，不仅需要有效可行的实验估计方法，还需要有评价模型泛化性能的评价标准，这就是指性能度量。在分类任务中，最常用的两种性能度量的指标就是准确度（Accuracy）和错误率（Error Ratio）。准确度是测试样本中被分类正确的样本数量和总体测试样本数量的比例，错误率是指在测试样本中被分类错误的样本占总体测试样本的数量的比例。

准确度和错误率这些指标虽然经常使用但是还是无法满足所有任务的需求。对于二元分类问题，可以将预测结果（真，假）和真实标签结果（真，假）组合为四种不同的情况：真正例（TP，预测是真，标签也是真）、假正例（FP预测为假，但是标签为真）、真反例（TN，预测为假，标签也是假）和假反例（FN，预测为假但是标签为真）。对于评价指标来说，我们同时也会注重查全率（Precision）和召回率（Recall）等度量指标。查全率的意义是：它计算的是所有"正确被检索的(TP)"占所有"实际被检索到的(TP+FP)"的比例。召回率的意义是：它计算的是所有"正确被检索的(TP)"占所有"应该检索到的(TP+FN)"的比例。一般来说，查全率高的情况下召回率就会偏低，召回率高的情况下查全率就会偏低，两者有着矛盾的关系，因此，需要一个指标将这两个不同的指标同时考虑进去，而这个指标在二元分类上就是F1得分。

查全率： 

召回率：

|  |  |  |
| --- | --- | --- |
| 真实情况 | 预测结果 | |
| 正例 | 反例 |
| 正例 | TP（真正例） | FN（假反例） |
| 反例 | FP（假正例） | TN（真反例） |



1. 分类模型（好听的话还要自己扩充）

我们使用由谷歌最新开发的TensorFlow-2.0来开发我们所需要的分类模型。Tensorflow-2.0集成了Keras的高级API接口。因此，在本次实验中，我们主要使用Kears的函数高级接口和TensorFlow的低级函数接口组合开发本次实验所需要的模型。

在所有的数据中，标签为0的数据有28179条，但是标签为1的数据有16585条。两个样本的数据量之间的比值约为1.699。在如此样本不平衡的状态下进行训练可能会使分类模型偏向于预测数据量多的那一类数据，因此，我们需要进行样本平衡策略。

我使用的策略是：在损失函数中对每个类数据的损失提供不同的权重，不同类别因为类别权重的不同在训练反向传播时的梯度也不一样，以此来平衡数据不平衡带来的缺陷。0标签类的数据权重为0.6165；1标签类的数据权重为1.0。

我通过随机数的方法，将数据集随机的划分为训练集合和测试集合。训练集合上的数据用来训练我们的分类模型，测试集上的数据用来对分类模型的效果进行度量。两者之间的交集为空集，两者之间的并集为整个数据集合。由于数据量很大，太多的测试集合可能会花费很长的时间，因此，我们使用44844个样本的数据作为训练集合，453个样本的数据作为测试集合。

**（这里写对数据的预处理部分：清洗，分词，词向量化）**

通常来说，CPU是电脑的计算核心。但是，CPU是顺序计算单元，在并行计算上的运算速度很慢。GPU是显示图形单元，由于有多达上百万，千万的像素点需要计算处理，所有，GPU的并行处理速度很快，但是顺序处理速度较慢。

在深度学习中，模型复杂，数据量大，仅仅使用CPU作为计算单元的处理对于庞大的计算量来说是远远不够的，因此，需要使用GPU并行计算单元来加速计算速度。

本实验全程采用GPU加速单元加速模型的训练以及测试。GPU所使用的是GTX1080ti。

**（四种模型介绍，LSTM,Bi-LSTM,Bi-LSTM-Attention,Transformer，可以介绍详细点）**

1. 网络的训练（好听的话还要自己扩充）

Keras框架中Model类的fit函数可以直接把你写好的模型拿去训练，训练的大部分参数并不需要手工指定，Keras会自动帮你设置。在fit函数中，需要提供训练数据，训练数据的标签，测试集合的数据和标签这四样东西即可。在模型的编译阶段，需要提供训练整个网络的优化器以及对应的损失函数。

### 优化器的选择（好听的话还要自己扩充）

在训练网络的过程中，我们使用带动量（momentum）和牛顿法（nesterov）的SGD（随机梯度下降）优化器。在多篇论文中提到，自适应优化器（代表Adam）在很多应用场景下的泛化性能会低于传统SGD 2~3个百分点（可以百度出来哪篇论文，作为引用）。

自适应优化器的优点是学习率是根据数据进行动态调整的，因此收敛速度极快，但是容易优化到局部最小值的点，因此导致模型的泛化性能不强，并且在某些数据下可能导致无法收敛。传统的SGD优点是泛化性能在精调下比自适应性优化器强，可以在学习率下降的情况下收敛，但是收敛速度慢，可能会被困在局部的鞍点之中。

带动量的SGD在一定程度上可以帮助SGD越过鞍点的限制，加速收敛速度，并且带牛顿法的动量SGD可以通过二阶海森矩阵（Hessian Matrix）估计当前时刻的局部曲率，为下一次梯度的更新提供一定程度上的估计，防止模型偏离正常梯度。

### 损失函数的选择（好听的话还要自己扩充）

### KL散度

对于分类问题，我们得到的最后结果应该是一个概率值（介于0~1之间）而不是一个回归值（R实域）。对于预测概率分布与真实标签概率分布的度量，我们首先想到的是KL散度，其公式如下：



P是真实标签分布，Q是预测的分布，N为样本数量。如果两个分布差异越大，那么KL散度就会越大。

### 交叉熵

交叉熵损失就是KL散度的一个简化版本，它的公式如下，相对于MSE损失来说，交叉熵损失可以在一定程度上解决梯度消失的问题：



同样评价的是预测概率和真实概率之间的差距大小。

### 二元交叉熵

由于我们的模型是二分类，可以使用对应于二分类特别的交叉熵损失，其公式如下：



ti为概率预测值（0~1之间），yi为真实标签。、

### 均方差（MSE）

MSE损失如下：



MSE的损失在训练过程中可能产生梯度消失的问题，因此，我们不采用做回归的MSE损失函数。

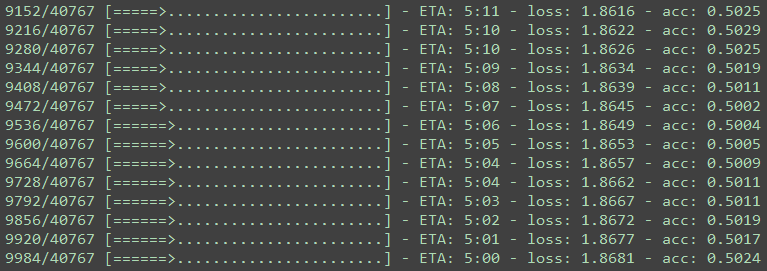
所以，最终，我们选用二分类的交叉熵损失函数作为我们评价分类模型的预测值和真实标签之间距离的损失函数。

（L2正则化的原理可以百度）

在损失函数上我们还会加入L2正则化项。L2正则化可以较为完美的防止模型的过拟合。由于数据量庞大，并且带一定程度的噪声，因此，我们将L2正则化的权重调小，其值为5e-6。

### 训练过程 （好听的话还要自己扩充）

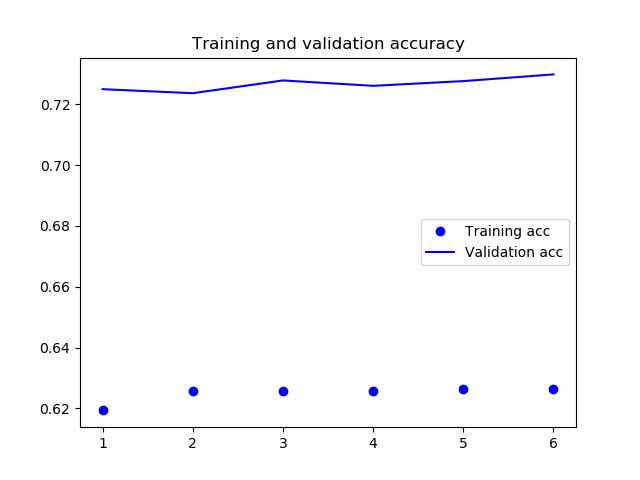
使用带动量的SGD和二交叉熵损失训练分类模型。训练总体训练集合6次，每64个训练样本一个迷你批次。



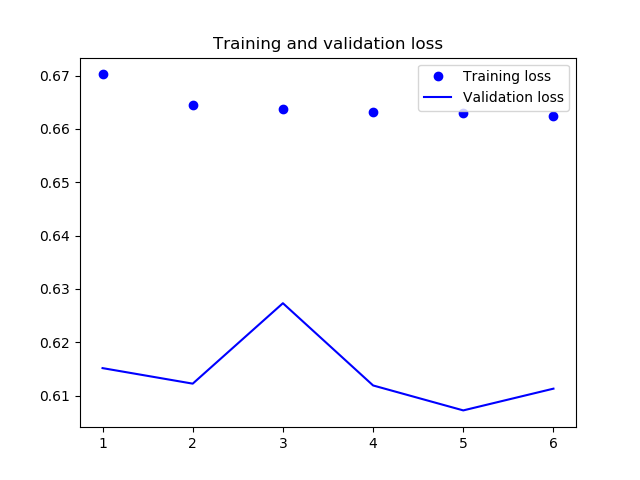
训练时间大约是每10分钟一个轮回，6轮需要大约60分钟。

1、LSTM

LSTM的训练过程中，训练集上的准确度和测试集上的准确度如图所示：

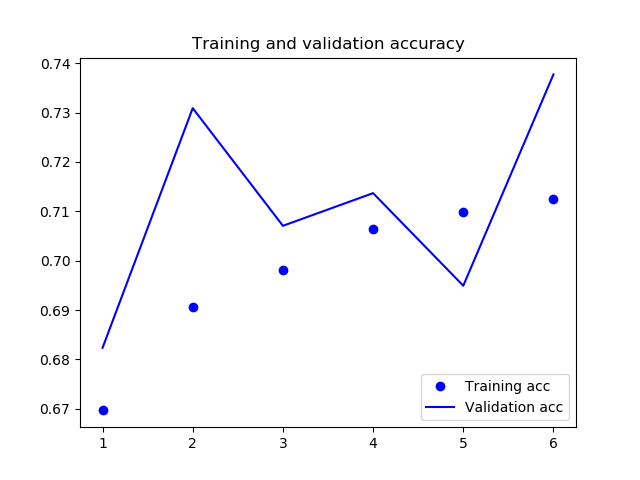


在训练集上的损失和测试集合上的损失如图所示：

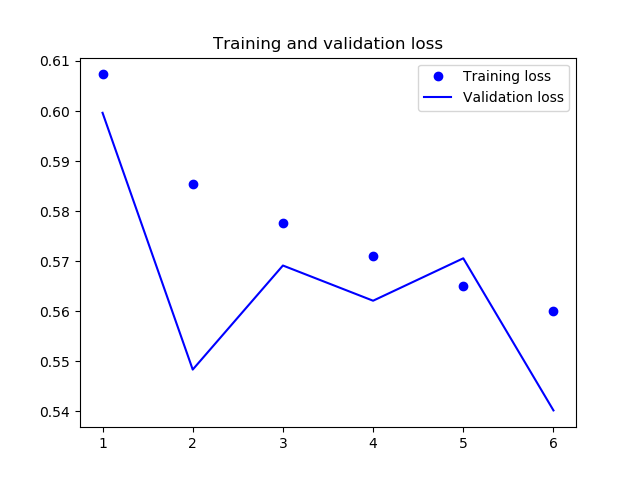


1. Bi-LSTM

Bi-LSTM的训练过程中，训练集上的准确度和测试集上的准确度如图所示：

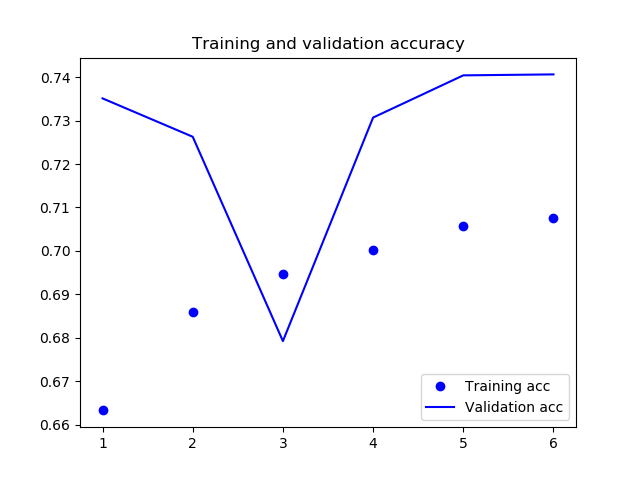


在训练集上的损失和测试集合上的损失如图所示：

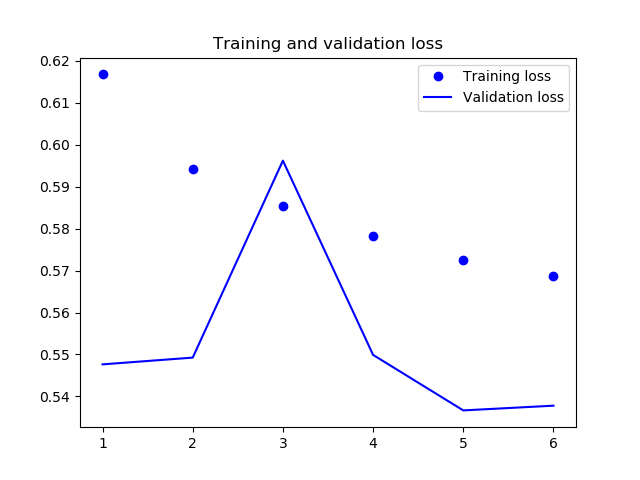


1. Bi-LSTM-Attention

Bi-LSTM-Attention的训练过程中，训练集上的准确度和测试集上的准确度如图所示：

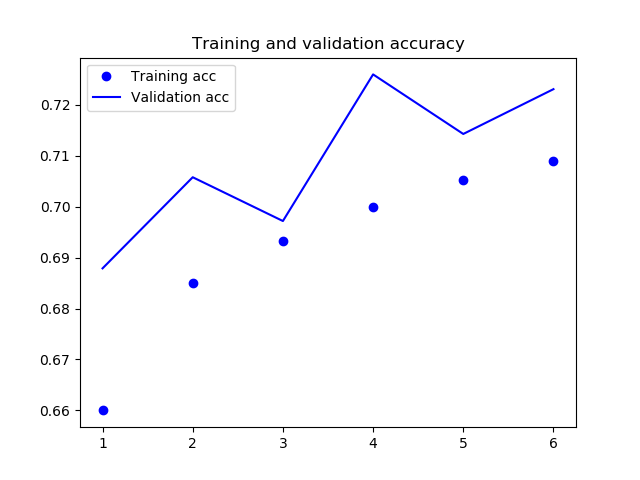


在训练集上的损失和测试集合上的损失如图所示：

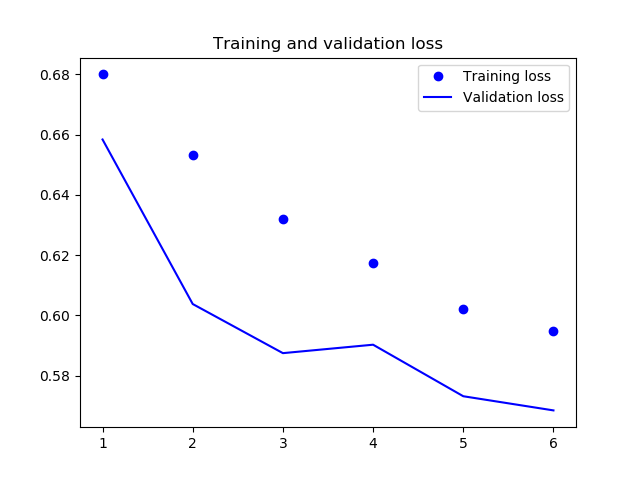


1. Transformer

Transformer的训练过程中，训练集上的准确度和测试集上的准确度如图所示：



在训练集上的损失和测试集合上的损失如图所示：



1. 训练结果（一些好听的话还要自己扩充）

在对模型完成训练之后，我们需要对我们的模型进行评价。本次实验使用同一文本数据训练了4种不同的模型，它们分别是LSTM、双向LSTM、双向LSTM加上注意力机制（Attention）和Transformer模型。

在确定了训练好的模型和度量的指标之后，就需要使用训练好的模型对测试样本进行测试了并且使用不同的度量指标对测试样本的测试结果进行评估，评估结果如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 准确度 | F1值 | 查全率 | 召回率 |
| Transformer | 72.605% | 0.8412 | 0.7261 | 0.9978 |
| LSTM | 72.980% | 0.8425 | 0.7304 | 0.9951 |
| Bi-LSTM | 73.775% | 0.8317 | 0.7787 | 0.8924 |
| Bi-LSTM-Attention | 74.062% | 0.8282 | 0.7977 | 0.8611 |

通过上标提供的结果来看，与随机猜测的准确度50%相比，我们的模型可以达到74.062%的准确率，相比于50%的猜测结果高很多。因此证明有很好的效果。

但是，由于数据来自于微博，有很大的噪声，并且，标签不一定准确，因此，如果还想提升模型的准确度还需要从数据的来源，数据的处理和确定标签的正确性下手，提高数据的质量。