# 深度学习与自然语言处理第二次作业报告

吴宣余

ZY2303121

wxy7334@buaa.edu.cn

## 一、问题描述

从给定的语料库中均匀抽取1000个段落作为数据集，每个段落的标签对应段落所属的小说。利用LDA模型在给定的语料库上进行文本建模，并把每个段落表示为主题分布后进行分类，验证与分析分类结果。

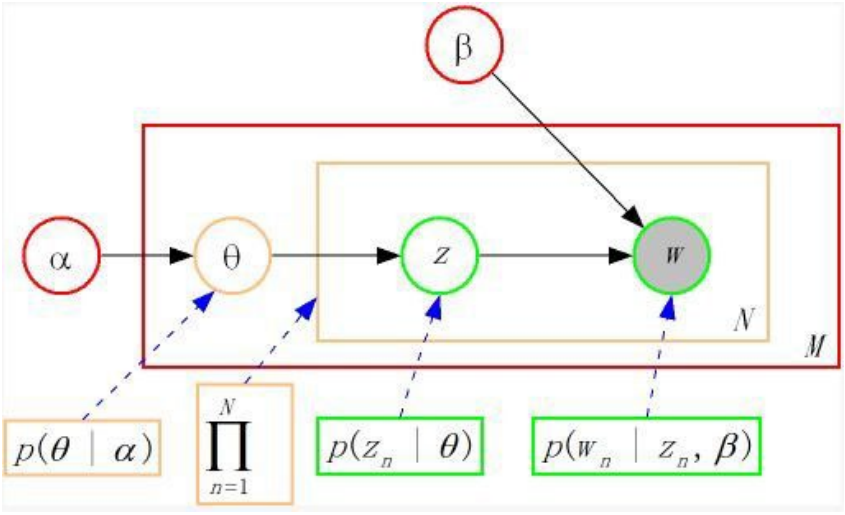
## 二、LDA模型

在文本挖掘领域中大量的数据都是非结构化的，难以从信息中直接获取相关和期望的信息。主题模型（Topic Model）能够识别在文档里的主题，并且挖掘语料里隐藏信息，在主题聚合、特征选择等场景有广泛的用途。

LDA（LinearDiscriminantAnalysis）是一种文档主题生成模型，它可以将文档中每篇文档的主题按照概率分布的形式给出。也被称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。所谓生成模型，即认为一篇文章的每个词都是通过“以一定概率选择了某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。

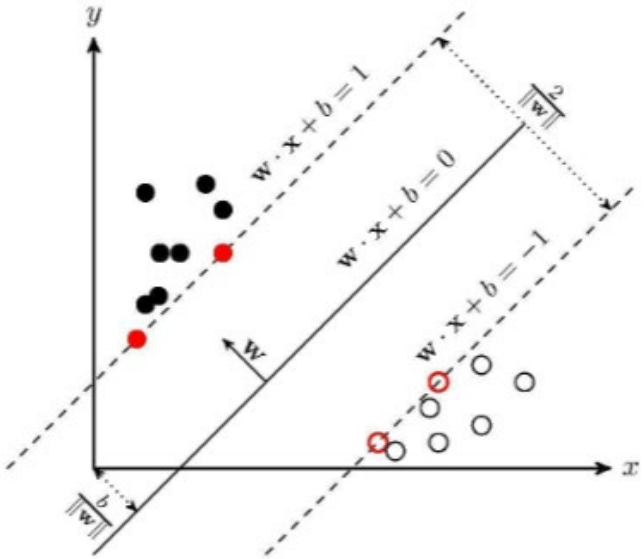
LDA是一种非监督机器学习技术，可以用来识别大规模文档集（documentcollection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，该方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

LDA的核心思想是寻找到最佳的投影方法，将高维的样本投影到特征空间(feature space)，使得不同类别间的数据“距离”最大，而同一类别内的数据“距离”最小。LDA采用了“词袋”的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。




## 三、SVM分类器

SVM学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示， wx+b=0 即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。支持向量机（SVM）是具有相关学习算法的监督学习模型，其分析用于分类和回归分析的数据。给定一组训练示例，每个示例标记为属于两个类别中的一个或另一个，SVM训练算法构建一个模型，将新示例分配给一个类别或另一个类别，使其成为非概率二元线性分类器。SVM模型是将示例表示为空间中的点，映射使得单独类别的示例除以尽可能宽的明确间隙。然后将新的示例映射到同一空间，并根据它们落在哪个边缘预测属于一个类别。



## 四、实验

1. 数据预处理与与数据集划分

从给定的16本小说语料库中均匀抽取1000个段落作为数据集，每个段落取500个token数，每个段落的标签为对应的小说名。同时对抽取到的每个段落进行删除换行符、分页符等隐藏符号；删除标点符号；删除广告信息等yu处理。对预处理后的数据集进行随机划分，其中90%作为训练集，10%作为测试集。

```
def preprocess(corpus):
    pattern = r'[^\u4E00-\u9FA5]'
    regex = re.compile(pattern)
    replacements = ["\t", "\n", "\u3000", "\u0020", "\u00A0", " "]
    corpus = rid_of_ad(corpus)
    corpus = re.sub(regex, "", corpus)
    for replacement in replacements:
        corpus = corpus.replace(replacement, "")
    return corpus


def ReadData(path):
    content = []
    names = os.listdir(path)
    FileNum = len(names)
    ParaLength = 500 #每段长度
    ParaNum = 1000
    SelectNum = (ParaNum//FileNum) + 1
    for name in names:
        NovelName = path + '\\' + name
        with open(NovelName, 'r', encoding= 'ANSI') as f:
            con = f.read()
            con = preprocess(con)
            con = jieba.lcut(con)
            selectPos = len(con)//SelectNum
            for i in range(SelectNum):
                SelectStart =  random.randint(selectPos*i, selectPos*(i+1))
                para = con[SelectStart:SelectStart+ParaLength]
                content.append((name,para))
        f.close()
    content = content[:ParaNum]
    return content
```

2. 创建LDA模型并训练SVM分类器

利用训练样本训练LDA模型，其中主题数为 T ，每个训练样本对应的主题分布为一个$1 \times T$的向量。利用上述训练样本特征向量以及对应标签训练一个线性SVM分类器；并预测训练样本的标签，得到训练准确率。

```
def LDA(train_data, train_label, test_data, test_label,num_topics = 2000):
    dictionary = corpora.Dictionary(train_data)
    lda_corpus_train = [dictionary.doc2bow(tmp_doc) for tmp_doc in
train_data]
    lda = models.LdaModel(corpus=lda_corpus_train, id2word=dictionary,
num_topics = num_topics)

    #### train svm classifier for correct label
    train_topic_distribution = lda.get_document_topics(lda_corpus_train)
    train_features = np.zeros((len(train_data), num_topics))
    for i in range(len(train_topic_distribution)):
        tmp_topic_distribution = train_topic_distribution[i]
        for j in range(len(tmp_topic_distribution)):
            train_features[i][tmp_topic_distribution[j][0]] =
tmp_topic_distribution[j][1]

    assert len(train_label) == len(train_features)
    train_label = np.array(train_label)
    classifier = SVC(kernel='linear', probability=True)
    classifier.fit(train_features, train_label)
    print("训练集准确率:
{:.4f}.".format(sum(classifier.predict(train_features) == train_label) /
len(train_label)))

    lda_corpus_test = [dictionary.doc2bow(tmp_doc) for tmp_doc in test_data]
    test_topic_distribution = lda.get_document_topics(lda_corpus_test)
    test_features = np.zeros((len(test_data), num_topics))
    for i in range(len(test_topic_distribution)):
        tmp_topic_distribution = test_topic_distribution[i]
        for j in range(len(tmp_topic_distribution)):
            test_features[i][tmp_topic_distribution[j][0]] =
tmp_topic_distribution[j][1]
    assert len(test_label) == len(test_features)
    test_label = np.array(test_label)
    print("测试集准确率:
{:.4f}.".format(sum(classifier.predict(test_features) == test_label) /
len(test_label)))
```

3. 测试样本的主题分布

利用训练好的LDA模型得到测试样本的主题分布。计算测试样本文本分类准确率。

```
if __name__ == '__main__':
    context = ReadData('dataset')
    traindata, trainlabel, testdata, testlabel = Dataset(context)
    LDA(traindata, trainlabel, testdata, testlabel)
```

## 五、实验结果与分析

按如下参数生成文本数据集：

| 参数 | 文本数 | Token 数 | 段落数 |
|---|---|---|---|
| 数值 | 16 | 500 | 1000 |

1. 改变LDA模型主题个数T，得到的分类结果如下所示：

| 主题个数T | 训练准确率 | 测试准确率 |
|---|---|---|
| 50 | 0.2667 | 0.1800 |
| 100 | 0.2822 | 0.2100 |
| 200 | 0.3256 | 0.2300 |
| 500 | 0.4522 | 0.2600 |
| 1000 | 0.5889 | 0.3300 |
| 2000 | 0.6756 | 0.4100 |

2. 在原来的LDA模型的基础上，取主题个数T为100，分别以字和词为基本单元进行实验，得到的分类结果如下所示：

| 分词方式 | 训练准确率 | 测试准确率 |
|---|---|---|
| 词 | 0.2822 | 0.2100 |
| 字 | 0.2463 | 0.2000 |

3. 在原来的LDA模型的基础上，取主题个数T为100，改变段落的token数，得到的分类结果如下所示：

| token数 | 训练准确率 | 测试准确率 |
|---|---|---|
| 50 | 0.2500 | 0.1200 |
| 100 | 0.2519 | 0.1300 |
| 200 | 0.2544 | 0.1300 |
| 500 | 0.2667 | 0.2100 |
| 1000 | 0.2711 | 0.2400 |
| 2000 | 0.2800 | 0.3100 |

分析：

通过实验可以看出：随着主题个数的增加，训练准确度和测试准确度在增高，但越往后增加越不明显；且一般来说以词为基本单元得到的分类准确率要高于以字为基本单元，因此分类效果按词分割大于按字分割，说明按词分割更加适合LDA 主题提取，样本有更多的信息得到保留；随着段落 token 数的增加，分类准确率在缓慢上升，但是上升的效果不是很明显，说明分类的准确率主要由 LDA 模型的主题个数决定。