**《机器学习》实验指导书**

广州理工学院

# 《机器学习》实验五 隐马尔可夫模型应用实验

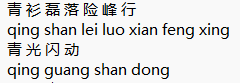
## 一、实验目的

1）掌握隐马尔可夫模型的应用方法。

2）掌握序列标注问题的基本处理方法。

## 二、实验内容

1）本次实验的具体内容是模拟全拼输入法。老师提供的训练语料（以小说《天龙八部》生成）包括许多以句子为单位的中文序列和对应的全拼拼音序列组成的训练样本，训练样本示例如下：



2）对训练样本进行统计，得到HMM模型要求的初始概率、转移概率和观测概率，实例化hmmlearn.hmm模型，对测试样本进行预测。

3）在提供的hmm\_model.ipynb文件中填写代码，完成指定任务。

## 三、实验参考

**1.全拼输入法**

全拼输入法的基本功能是将键盘输入序列映射到文字序列，例如键盘按下“ni hao”这六个键，输出汉字“你好”。

由于同音字和同音词等现象的存在，无法建立字母串和汉字词的一一对应关系表，因此，需要根据上下文来判断输入的字母串应该是哪一个汉字词，如“和”和“合”、“简洁”和“简捷”等。

因此，如何利用上下文来确定正确的汉字词是拼音输入法的关键问题。该问题属于序列标注问题，因此可以采用隐马尔可夫模型、条件随机场模型和循环神经网络模型等来解决。

确定了合适的模型类型和结构之后，还需要有大量的样本来训练模型，才能得到一个正确率高的输入法模型。

**2.隐马尔可夫模型**

隐马尔可夫模型是关于时间序列的概率模型，描述由一个隐藏的马尔可夫链生成不可观测的状态序列，再由状态序列生成一个观测序列的过程，序列的每一个位置可以看作是一个时刻。它常用于解决标注问题。

隐马模型相关知识参考教材6.2小节。

**3.hmmlearn隐马尔可夫模型扩展库**

hmmlearn[[1]](#footnote-0)曾经是scikit-learn项目的一部分，现已独立成单独的Python扩展库。hmmlearn实现了三种HMM模型类，按照观测状态是连续状态还是离散状态，可以分为两类。GaussianHMM和GMMHMM是连续观测状态的HMM模型。CategoricalHMM是离散观测状态的模型，可以用于实现拼音输入法，其原型如下：

|  |
| --- |
| * class hmmlearn.hmm.CategoricalHMM( n\_components=1, # 隐藏状态的种类   # 以下参数用于模型训练：  n\_trials=None,  startprob\_prior=1.0, # 由模型本身对初始概率分布随机初始化  transmat\_prior=1.0, # 由模型本身对状态转移概率随机初始化  algorithm='viterbi', # 解码方法  n\_iter=10, # 训练迭代的次数  tol=0.01, # 训练结束的阈值  verbose=False, # 训练过程是否可见  params='ste', # 训练过程中更新的参数，默认为所有的参数  init\_params='ste', # 由模型进行随机初始化的参数，默认为所有的参数  implementation='log') # 选择对数/缩放进行前向或后向算法  ) |

## 四、实验要求

在提供的hmm\_model.ipynb文件中填写代码，完成以下指定任务（须运行后得到合理结果）。

1）编写代码完成对HMM模型三个概率的统计工作；

2）编写代码完成对三个概率矩阵的归一化工作；

3）缩写代码实例化CategoricalHMM类，并导入三个概率矩阵；

4）用模型对两个测试语句进行测试，观察结果并分析原因。

hmm\_model.ipynb文件如下所示，其中需要填写代码的地方用注释说明。

|  |
| --- |
| import numpy as np  # 从训练数据文件中读取样本和标签，并统计每个汉字和全拼拼音出现的次数。  # 训练数据文件中，前一行为文本样本，后一行为对应的拼音标签。可用记事本打开观察一下。  file = open("traindata.txt", encoding='utf-8')  char\_dict = {}  pinyin\_dict = {}  lines = file.readlines()  tag = True  for line in lines:  if tag:  temp\_dict = char\_dict  tag = False  else:  temp\_dict = pinyin\_dict  tag = True  for cent in line.split():  if cent in temp\_dict:  temp\_dict[cent] += 1  else:  temp\_dict[cent] = 1  char\_id = list(char\_dict) # 用列表的下标对汉字进行编号，可用list.index()获取对应汉字的编号，用list[i]获取对应编号i的汉字  pinyin\_id = list(pinyin\_dict) # 用列表的下标对拼音进行编号  # 定义初始概率矩阵pi、转移概率矩阵A和观测概率矩阵B  # 与教材上的处理方式不同，采用全1的平滑处理方式，该处理是符合hmmlearn.hmm的要求。  # 但是这种处理方式简单，仅适合学生入门，效果有时不好。  pi = np.ones(len(char\_id))  A = np.ones((len(char\_id), len(char\_id)))  B = np.ones((len(char\_id), len(pinyin\_id)))  # 统计3个概率矩阵  ### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ###  ### 编写代码完成3个概率矩阵的统计工作  # 对统计得到的3个概率矩阵归一化  ### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ###  ### 编写代码完成3个概率矩阵归一化  from hmmlearn import hmm  model = hmm.CategoricalHMM(n\_components=len(char\_id))  ### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ###  ### 编写代码导入3个概率矩阵  test = 'duan yu wen ni shi qiao feng hai shi xiao feng' # 因为是用《天龙八部》作为语料库，该句预测较好  #test = 'rong er shi she diao ying xiong chuan li de ren wu' # 该句预测较差  test = test.split()  test\_data = []  for i in range(len(test)):  test\_data.append(pinyin\_id.index(test[i]))  test\_data = np.array(test\_data).reshape(-1, 1)  test\_out = model.predict(test\_data)  test\_out\_sentence = ""  for id in test\_out:  test\_out\_sentence += char\_id[id]  print(test\_out\_sentence)  >>>段誉问你是乔峰还是萧峰 |

## 五、实验考核

编制的程序的正确运行（70%）+实验报告(30%)。

## 六、实验报告

实验结束后，撰写实验报告，实验报告【实验内容和原理】部分根据授课相关内容和实验指导书相关内容填写；【操作方法和实验步骤】部分应包括关键部分源代码及功能注释。【实验结果与分析】部分写清楚个人是如何分析实验内容将其转换为源代码，得到了什么样的运行结果，运行结果的含义是什么。

1. https://hmmlearn.readthedocs.io/en/latest/tutorial.html [↑](#footnote-ref-0)