实验6-3：隐马尔可夫模型

本次实验旨在让同学们了解并掌握隐马尔可夫模型的原理和其三个基本问题的求解方法。

内容

1. 马尔科夫链

考虑一个随机变量的序列 $X = X_0, X_1, \ldots, X(t), \ldots$ 这里 X_t，表示时刻 t 的随机变量， $t = 0, 1, 2....$ 每个随机变量 $X_t (t = 0, 1, 2, \ldots)$ 的取值集合相同，称为状态空间. 随机变量可以是离散的， 也可以是连续的。随机变量的序列构成随机过程（stochastic process)。

假设在时刻 0 的随机变量 X_0 遵循概率分布 $P(X_0) = \pi$，称为初始状态分布。在某个时刻 $t >= 1$ 的随机变量 X_t与前一个时刻的随机变量 X_{t-1} 之间有条件分布 $P(X_t|X_{t-1})$ 如果 X_t 只依赖于 X_{t-1}, 而不依赖于过去的随机变量 X_0 ，$X_1, \ldots$ ，X_{t-2} 这一性质称为马尔科夫性，即$P(X_t|X_0, X_1, \ldots, X_{t-1}) = P(X_t|X_{t-1})$， $t = 1, 2, \ldots$

具有马尔科夫性的随机序列$X = X_0, X_1, \ldots, X(t), \ldots$称为**马尔科夫链**， 或马尔科夫过程（Markov process)。 条件概率分布 $P(X_t|X_{t-1})$ 称为**马尔科夫链的转移概率分布**。

2. 隐马尔科夫模型（Hidden Markov Model）

隐马尔科夫模型（HMM）是经典的机器学习模型之一，它在语言识别、自然语言处理、模式识别等领域得到广泛的应用。当然，随着目前深度学习的崛起，尤其是RNN、LSTM等神经网络序列模型的火热，HMM的地位有所下降。但是作为一个经典的模型，学习HMM的模型和对应算法，对我们解决问题建模的能力提高以及算法思路的拓展还是很好的。

2.1 HMM的定义

设隐藏状态集合 Q 是所有可能的隐藏状态的集合$Q = \{q_1, q_2, \ldots, q_N\}$，$V$ 是所有可能的观测状态的集合，$V = \{v_1, v_2, \ldots, v_M\}$。其中，$N$ 是可能的隐藏状态数量，M 是所有的可能的观测状态数量。

对于一个长度为 T 的序列，对应的隐藏状态序列$I = \{i_1, i_2, \ldots, i_T\}$，和对应的观测序列$O = \{o_1, o_2, \ldots, o_T\}$。 其中，任意一个隐藏状态 $i_t \in Q$，任意一个观测状态 $o_t \in V$。

HMM有两个重要假设：

1) 齐次马尔科夫链假设。即任意时刻的隐藏状态只依赖于它前一个隐藏状态。当然这样假设有点极端，因为很多时候我们的某一个隐藏状态不仅仅只依赖于前一个隐藏状态，可能是前两个或者是前三个。但是这样假设的好处就是模型简单，便于求解。如果在时刻t的隐藏状态是$i_t = q_i$，在时刻$t + 1$的隐藏状态是$i_{t+1} = q_j$，则从时刻t到时刻$t + 1$的HMM状态转移概率可以表示为：$a_{ij} = P(i_{t+1} = q_j|i_t = q_i)$。这样$a_{ij}$可以组成马尔科夫链的状态转移矩阵$A = [a_{ij}]_{N\times N}$

2) 观测独立性假设。即任意时刻的观察状态只仅仅依赖于当前时刻的隐藏状态，这也是一个为了简化模型的假设。如果在时刻t的隐藏状态是$i_t = q_j$，而对应的观察状态为$O_t = v_k$，则时刻t观察状态v_k在隐藏状态q_j下的发生概率表示为$b_j(k)$,即：$b_j(k) = P(o_t = v_k|i_t = q_j)$ 这样$b_j(k)$可以组成观测状态生成的概率矩阵$B = [b_j(k)]_{N\times M}$

除此之外，我们需要一组在时刻$t = 1$的隐藏状态概率分布π：$\pi = [\pi(i)]_N$ 其中 $\pi(i) = P(i_1 = q_i)$

一个HMM模型，可以由隐藏状态初始概率分布π，状态转移概率矩阵A和观测状态概率矩阵B决定。即，HMM模型可以由一个三元组λ表示如下：$\lambda = (A, B, \pi)$

2.2 HMM的三个基本问题

HMM有三个经典的问题需要解决：

1) **概率计算问题**。即给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = o_1, o_2, \ldots, o_T$，计算在模型$\lambda$下观测序列出现的概率 $P(O|\lambda)$。这个问题的求解通常采用**前向-后向(forward-backward algorithm)算法**。

2) **学习问题**。即给定观测序列 $O = o_1, o_2, \ldots, o_T$，估计模型 $\lambda = (A, B, \pi)$ 的参数，使该模型下观测序列的条件概率 $P(O|\lambda)$ 最大。这个问题的求解需要用到基于EM算法的**鲍姆-韦尔奇(Baum-Welch)算法**。

3) **预测问题**，也称为解码问题。即给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = o_1, o_2, \ldots, o_T$，求最可能出现的对应的状态序列 I，这个问题的求解需要用到基于动态规划的**维特比(Viterbi)算法**。

3. hmmlearn库的使用

`hmmlearn` 以前是scikit-learn项目中的一部分，现在已经是一个单独的python包，更多信息可查询官方文档。

可用的模型:

- `hmm.CategoricalHMM` : 具有分类发射的隐马尔可夫模型，观测状态为离散的模型。
- `hmm.GaussianHMM` : 具有高斯发射的隐马尔可夫模型，观测状态为连续的模型。
- `hmm.GMMHMM` : 具有高斯混合发射的隐马尔可夫模型，观测状态为连续的模型。
- `hmm.MultinomialHMM` : 具有多项分布发射的隐马尔可夫模型，观测状态为离散的模型。
- `hmm.PoissonHMM` : 具有泊松发射的隐马尔可夫模型，观测状态为离散的模型。

常用模型参数：

- n_components : 隐藏状态数目
- covariance_type: 协方差矩阵的类型
- min_covar : 最小方差，防止过拟合
- startprob_prior : 初始概率向量
- transmat_prior : 转移状态矩阵
- algorithm : 所用算法
- random_state : 随机数种子
- n_iter : 最大迭代次数
- tol : 停机阈值
- verbose : 是否打印日志以观察是否已收敛

在python中直接使用下述命令进行安装：

In [ ]:
```
pip install hmmlearn
```

安装好后，我们可以直接在python代码中进行导入：

In [ ]:
```
from hmmlearn import hmm
```

3.1 向前-向后算法

这里我们以天气和行为的关系作例子。

设隐藏状态为：

In [ ]:
```
states = ["Rainy", "Sunny"]##隐藏状态
n_states = len(states)##隐藏状态长度
```

可能的观测状态为：

In [ ]:
```
observations = ["walk", "shop", "clean"]##可观察的状态
n_observations = len(observations)##可观察序列的长度
```

初始状态序列为：

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
start_probability = np.array([0.6, 0.4])##开始是Rainy和Sunny的概率
```

状态转移矩阵为：

In [ ]:
```
##天气转移混淆矩阵，即Rainy和Sunny之间的转换关系
transition_probability = np.array([
  [0.7, 0.3],
  [0.4, 0.6]
])
```

发射矩阵为：

In [ ]:
```
##天气和行为混淆矩阵，即Rainy和Sunny对应walk,shop,clean的概率
emission_probability = np.array([
  [0.1, 0.4, 0.5],
  [0.6, 0.3, 0.1]
])
```

构建一个MultinomialHMM模型：

In [ ]:
```
model = hmm.MultinomialHMM(n_components=n_states)#初始化模型
#对模型参数初始化
model.startprob_= start_probability #开始的转移概率
model.transmat_ = transition_probability #天气转换混淆矩阵
model.emissionprob_ = emission_probability #天气和行为混淆矩阵
model.n_trials = 1 # 假设每个时间点行为发生次数总和为1
```

给出一个观测序列：

In [ ]:
```
#已知行为序列
bob_Actions = np.array([2, 0, 1, 1, 2, 0])
O = np.eye(n_observations)[bob_Actions].astype(int)#将行为序列转换为独热编码
print(O)
```

```
[[0 0 1]
 [1 0 0]
 [0 1 0]
 [0 1 0]
 [0 0 1]
 [1 0 0]]
```

解决问题1，概率计算问题：已知模型和观测的行为序列，计算行为序列出现的概率。

结果为-6.892170869，其实得到的结果是以自然数e为底数ln(P) = -6.892170869，所以概率P = 0.00101570648021

In [ ]:
```
# 向前-向后算法
score = model.score(O, lengths=[len(bob_Actions)])
print(score)
```

```
-6.892170869202603
```

3.2 鲍姆-韦尔奇算法

解决问题2，学习问题：已知观测的行为序列，估计模型参数。

解决这个问题需要有一定的数据量，然后通过`model.fit()`方法来进行训练，自己生成一个模型，因此这里并不需要设置`model.startprob_`,`model.transmat_`,`model.emissionprob_`。

In [ ]:
```
model2 = hmm.MultinomialHMM(n_components=n_states, n_iter=1000, tol=0.01) #初始化模型
```

观测数据：

In [ ]:
```
X = np.array([[2, 0, 1, 1, 2, 0],[0, 0, 1, 1, 2, 0],[2, 1, 2, 1, 2, 0]])
OX = np.eye(n_observations)[np.ravel(X)].astype(int)#转换为独热编码
print(OX)
```

```
[[0 0 1]
 [1 0 0]
 [0 1 0]
 [0 1 0]
 [0 0 1]
 [1 0 0]
 [1 0 0]
 [1 0 0]
 [0 1 0]
 [0 1 0]
 [0 0 1]
 [1 0 0]
 [0 0 1]
 [0 1 0]
 [0 0 1]
 [0 1 0]
 [0 0 1]
 [1 0 0]]
```

训练模型：

In [ ]:
```
#鲍姆-韦尔奇算法
model2.fit(OX, lengths=[len(x) for x in X])
print("startprob:", model2.startprob_)
print("transmat:", model2.transmat_)
print("emissionprob:", model2.emissionprob_)
```

```
startprob: [9.99999515e-01 4.84821837e-07]
transmat: [[5.04498104e-07 9.99999496e-01]
 [4.99835000e-01 5.00165000e-01]]
emissionprob: [[1.42901617e-01 3.61934263e-09 8.57098379e-01]
 [4.54494092e-01 5.45414177e-01 9.17308880e-05]]
```

模型评分：

In [ ]:
```
score2 = model2.score(OX)
print(score2)
```

```
-16.104095364670574
```

可以进行多次fit，然后取评分最高的模型进行预测：

In [ ]:
```
# 预测最可能的隐藏状态
pre = model2.predict(O, lengths=None)
print(pre)
```

```
[0 1 1 1 0 1]
```

In [ ]:
```
# 预测各个隐藏状态的概率
prob = model2.predict_proba(O, lengths=None)
print(prob)
```

```
[[1.00000000e+00 4.22564111e-11]
 [3.17159209e-07 9.99999683e-01]
 [1.32587622e-08 9.99999987e-01]
 [9.39996403e-13 1.00000000e+00]
 [9.99929609e-01 7.03914237e-05]
 [1.69883257e-05 9.99983012e-01]]
```

3.3 维特比算法

解决问题3，解码问题：已知模型和观测的行为序列，估计最可能的天气。

In [ ]:
```
# 维特比算法
logprob, weathers = model.decode(O, algorithm="viterbi")
print("Bob Actions:", ", ".join(map(lambda x: observations[x], bob_Actions)))
print("weathers:", ", ".join(map(lambda x: states[x], weathers)))
```

```
Bob Actions: clean, walk, shop, shop, clean, walk
weathers: Rainy, Sunny, Rainy, Rainy, Rainy, Sunny
```

评估模型拟合的好坏，数值越大越好：

In [ ]:
```
from math import exp
print(exp(logprob))
```

```
0.0001524095999999999
```

4. 动手实践

隐马尔可夫模型（HMM）是一种统计模型，用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数，然后利用这些参数来作进一步的分析。

在**拼音输入法**中，输入法根据用户输入的拼音序列转换为对应的汉字，完成中文的输入。这里，输入的拼音序列就是一个可观察到的序列，而拼音对应的汉字则可以看做是隐藏序列。

因此，我们可以使用HMM通过观察变量求解隐藏变量，实现简单的拼音输入法：

1) 对语料库中的所有词按照单字分词，并统计各个词出现的频率，即可作为初始化概率π

2) 统计每个汉字后面出现的汉字的次数，以此作为隐藏状态的转移概率分布A

3) 统计每个拼音对应汉字以及各自出现的次数，就可以得到观察概率分布B

经过以上步骤之后，就能得到一个隐马尔科夫模型。**如果数据量过小，基于上面统计的方法可能会有概率为0的情况，解决办法可以是设定一个比较小的值(如1e-10)代替0来避免。**

In [ ]:
```
import re
import pypinyin #调用pypinyin库对语料进行注音
import numpy as np
from hmmlearn import hmm

class pinyin(object):
```

```
    def __init__(self):
        """
        初始化函数
        """
        self.pi = {}  # 初始状态概率 {word:pro, ..}
        self.A = {}  # 状态转移概率   {phrase:pro, ..}
        self.B = {}  # 观测状态概率 {pinyin:{word:pro, ..}, ..}
        self.dic = {}  # 拼音字典 {pinyin: wordwordword}
        self.word2id = {} # 汉字到id的映射
        self.id2word = {} # id到汉字的映射
        self.state_index = {} # 状态（汉字）的索引映射
        self.observation_index = {} # 观测（拼音）的索引映射


    def train_A_B_pi(self):
        """
        求初始状态概率pi、状态转移概率A、观测状态概率B
        :return:
        """
        traindata_path = r"DataSet/toutiao_cat_data.txt" # 语料数据集路径
        f = open(traindata_path, encoding='utf-8')
        # 统计单字、词组频数
        sw = {}
        dw = {}
        num_word = 0
        for line in f.readlines():
            line = re.findall('[\u4e00-\u9fa5]+', line)
            for sentence in line:
                pw = ""
                for word in sentence:
                    # 统计汉字到id的映射
                    if word not in self.word2id:
                        self.word2id[word] = num_word

                    # 统计单字
                    if word not in sw:
                        sw[word] = 1
                    else:
                        sw[word] += 1

                    # 统计词组
                    if pw != "":
                        if pw + word not in dw:
                            dw[pw + word] = 1
                        else:
                            dw[pw + word] += 1
                    pw = word
                num_word += 1
        f.close()

        # self.pi # 初始状态概率 {word:pro, ..}
        for word in sw.keys():
            self.pi[word] = sw[word] / num_word

        # self.A # 状态转移概率   {phrase:pro, ..}
        for phrase in dw:
            self.A[phrase] = dw[phrase] / sw[phrase[0]]

        # self.B 观测状态概率 {pinyin:{word:pro, ..}, ..}
        # 计算观测状态频数
        for word in sw:
            # lazy_pinyin 去除拼音平仄
            pinyin = pypinyin.lazy_pinyin(word)[0]
            if pinyin not in self.B.keys():
                self.B[pinyin] = {word: sw[word]}
            else:
                self.B[pinyin][word] = sw[word]
        # 计算观测状态频率
        for pinyin in self.B.keys():
            sum_word = sum(self.B[pinyin].values())
            for word in self.B[pinyin]:
                self.B[pinyin][word] = self.B[pinyin][word] / sum_word

        # self.dic # 拼音字典 {pinyin: wordwordword}
        dic_path = r"DataSet/pinyin2hanzi.txt" # 拼音字典路径
        f = open(dic_path, encoding='utf-8')
        for line in f.readlines():
            line = re.sub(r'[\ufeff]', '', line).strip().split()
            self.dic[line[0]] = line[1]
        f.close()

        # 翻转拼音字典，以便将拼音映射回汉字
        pinyin_to_words = {}
        for pinyin, words in self.dic.items():
            for word in words:
                if pinyin not in pinyin_to_words:
                    pinyin_to_words[pinyin] = []
                pinyin_to_words[pinyin].append(word)

        # id到汉字的映射
        self.id2word = { idx:state for idx, state in enumerate(self.word2id.keys())}
```

```
def buildHMM(self):
    """
    求HMM的start_prob、 transmat_prob、emission_prob
    :return:    modelpinyin
    """
    # 状态（汉字）、观测（拼音）的索引映射
    self.state_index = {state: idx for idx, state in enumerate(self.word2id.keys())}
    n_states = len(self.state_index)
    self.observation_index = {obs: idx for idx, obs in enumerate(self.dic.keys())}
    n_observations = len(self.observation_index)

    ## TODO: 求出start_prob、 transmat_prob、emission_prob，构建HMM模型，返回 modelpinyin
    raise NotImplementedError("此部分需要同学们自行实现。")


def cal_accuracy(self, text, label):
    """
    计算准确性
    :param text: 预测文本
    :param label: 真实文本
    :return: 准确性
    """
    num_right = 0
    for i in range(len(text)):
        if text[i] == label[i]:
            num_right += 1
    return num_right / len(text)
```

```
if __name__ == '__main__':
    pinyin = pinyin()
    pinyin.train_A_B_pi()
    modelpinyin = pinyin.buildHMM()
    print(modelpinyin.startprob_)
    print(modelpinyin.transmat_)
    print(modelpinyin.emissionprob_)

    # 读取测试文件数据
    test_path = r"DataSet/test1.txt"
    f = open(test_path, encoding='gbk')
    flag = True
    pinyin_list = []
    label_list = []
    for line in f.readlines():
        if flag:
            pinyin_list.append(line.strip('\n').lower())
        else:
            label_list.append(line.strip('\n'))
        flag ^= 1
    # 测试模型
    acc_sum = 0
    for i in range(len(pinyin_list)):
        # 将一行拼音序列转换为列表
        py_list = pinyin_list[i].lower().strip().split()
        # 将拼音序列转换为观测序列
        py_observations = np.array([pinyin.observation_index[py] for py in py_list])

        ## TODO: 使用维特比算法预测句子，返回 pre_words
        raise NotImplementedError("此部分需要同学们自行实现。")

        # 计算准确性
        acc = pinyin.cal_accuracy(pre_words, label_list[i])
        acc_sum += acc
        print(f"pinyin={pinyin_list[i]}")
        print(f"predict={pre_words}")
        print(f"label={label_list[i]}")
        print(f"accuracy={acc}\n")

    # 打印总的准确性
    print(f"average accuracy = {acc_sum / len(pinyin_list)}")
```

根据给出的代码框架，使用hmmlearn库实现拼音输入法：

- 求HMM的start_prob、transmat_prob、emission_prob，补全框架中HMM模型构建相关内容；
- 补全框架中维特比算法求解的内容，对test1.txt中的拼音序列进行预测；
- 临近期末 本次实验不要求提交实验报告，请同学们简单写一下对《机器学习》这门课程的心得或者意见建议并提交。