**实验2. 隐马尔科夫模型实践**

学号DZ1733017，姓名：孙锐，邮箱：450976770@qq.com

2017年12月2日

**综述**

机器学习最重要的任务是根据一些已观察到的事实（如训练样本）来对感兴趣的未知变量（例如类别标记）进行估计和推测。概率模型（probabilistic model）提供了一种描述框架，讲学习任务归结于计算变量的概率分布。在概率模型中，利用已知变量推测未知变量的分布称为“推断”（inference），其核心是如何基于可观测变量推断出未知变量的条件分布。

概率图模型（probabilistic graphical model）是一类用图来表达变量相关关系的概率模型。它以图为表示工具，最常见的是用一个结点表示一个或一组随机变量，结点之间的边表示变量间的概率相关关系，即“变量关系图”。根据边的性质不同，概率图模型大致可以分为两类：第一类是使用有向无环图表示变量间的依赖关系，称为有向图模型或者贝叶斯网（Bayesian network）；第二类是使用无向图表示变量间的相关关系，称为无向图模型或马尔可夫网（Markov network）。

隐马尔可夫模型（Hidden Markov Model，简称HMM）是结构最简单的动态贝叶斯网（dynamic Bayesian network），这是一种著名的有向图模型，主要用于时序数据建模，在语音识别、自然语言处理等领域有重要应用。

隐马尔可夫模型中的变量可以分为两组。第一组是状态变量，假定状态变量是隐藏的、不可被观测的，因此状态变量亦称为隐变量（hidden varible）。第二组是观测变量，其中表示在第i时刻的观测值。

本实验我们需要做的事是，首先模型化观测变量和状态变量的个数和取值，然后对数据进行模型化，得出观测序列；接着采用前向后项算法对观测序列进行训练，得到状态转移矩阵和发射矩阵以及初始状态分布，最后利用维特比算法根据训练的参数得到一个最有可能状态路径。作为应用，我们将设计好的模型来预测股票走势，检验模型在某只股票的历史数据上的准确率。

**任务1. 维特比算法**

对于一个已经实现好的隐马尔科夫模型，我们可以实现一个维特比算法，用动态规划的思想对模型进行推断。在该算法中，我们有如下参数（最后一个为输出）：

（1）状态空间，即隐变量空间，在这个例子中为，其中0表示熊市，1表示牛市，因此状态数N=2。

（2）观测量序列，T为样本数，在这个例子中为，其中0：跌，1：涨，2：平，因此观测状态数K=3。

（3）a是状态转移矩阵（transition matrix），大小为N\*N，a[i,j]表示从状态到的状态转移概率。

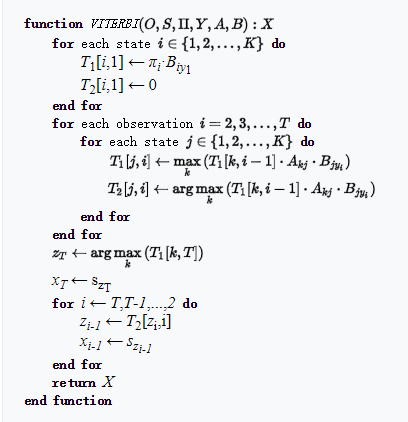
（4）b是发射矩阵（emission matrix），大小为N\*K，b[i,j]表示从状态观测到观测量的概率。

（5）pi是初始分布（initial probabilities），

（6）输出是最有可能的隐变量序列path（0,1序列）。

其中矩阵a，b和序列pi是一个训练好的HMM提供的参数。

算法用下图伪代码简略表示：



用python实现以上算法为：

global s

for i in xrange(N):

delta[i,0] = pi[i] \* b[i,o[0]]

phi[i,0] = 0

for t in xrange(1,T):

for i in xrange(N):

for k in xrange(N):

if delta[i,t] < (delta[k,t-1]\*a[k,i]\*b[i,o[t]]) :

delta[i,t] = delta[k,t-1]\*a[k,i]\*b[i,o[t]]

phi[i,t] = k

z = np.zeros(T)

m = 0

for k in xrange(N):

if m < delta[k,T-1] :

m = delta[k,T-1]

path[T-1] = k

for i in xrange(T-1,1,-1):

path[i-1] = phi[int(path[i]),i]

return path

**任务2. 实现 Forward Algorithm**

本过程我们需要实现一个N\*T的数组alpha，alpha(i,t)表示在观察到时，t时刻系统状态的概率。首先对alpha第一列进行初始化(pi为初始状态分布)：

对数组alpha逐列进行更新：

加上归一化因子，使得数组中每一列的和归一化为1，这样

因此，可以算出前向概率：

用python实现以上算法为：

st = 1

global s

s = np.zeros(T)

for i in xrange(N):

s[0] = s[0] + alpha[i,0]

for t in xrange(1,T):

c = np.diag(b[:,o[t]])

alpha[:,t] = alpha[:,t-1] .dot(a) .dot(c)

for i in xrange(N):

s[t] = s[t] + alpha[i,t]

for t in xrange(T):

st = st \* s[t]

alpha[:,t] = (1./st) \* alpha[:,t]

return alpha

**任务3. 实现 Backward Algorithm**

本过程我们同样需要实现一个N\*T的数组beta，beta(i,t)表示当系统状态时，在t时间之后出现观测序列的概率。本算法对beta逐列从后开始更新，因此设置初始状态为，后续有：

加上前向算法中算出的归一化因子有：

所以最终有：

因此系统在给定时刻处在任意状态的概率可以给出：

用python实现以上算法为：

st = 1

for t in xrange(T-2,0,-1):

d = np.diag(b[:,o[t+1]])

beta[:,t] = a .dot(d) .dot(beta[:,t+1])

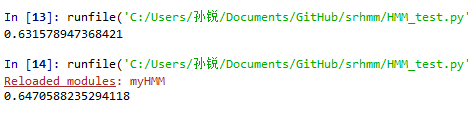
st = st \* s[t+1]

beta[:,t] = (1./st) \* beta[:,t]

return beta

**实验结果：**

实现算法后，运行命令pip install Tushare，安装股票数据导入模块，然后在集成编辑软件Spyder下运行文件Hmm\_test.py，得到如下结果In[13]，将num\_tests=N/test\_window命令改为num\_tests=N//test\_window得到结果In[14]:



**参考文献：**

【1】周志华. 机器学习. 清华大学出版社. 2016

【2】https://en.wikipedia.org/wiki/Viterbi\_algorithm

【3】https://en.wikipedia.org/wiki/Forward-backward\_algorithm

【4】https://en.wikipedia.org/wiki/Baum-Welch\_algorithm