马尔可夫过程：

马尔可夫过程（Markov process）是一类随机过程。它的原始模型马尔可夫链，由俄国数学家A.A.马尔可夫于1907年提出。该过程具有如下特性：在已知目前状态（现在）的条件下，它未来的演变（将来）不依赖于它以往的演变（过去 ）。例如森林中动物头数的变化构成——马尔可夫过程。在现实世界中，有很多过程都是马尔可夫过程，如液体中微粒所作的布朗运动、传染病受感染的人数、车站的候车人数等，都可视为马尔可夫过程。

每个状态的转移只依赖于之前的n个状态，这个过程被称为1个n阶的模型，其中n是影响转移状态的数目。最简单的马尔可夫过程就是一阶过程，每一个状态的转移只依赖于其之前的那一个状态，这个也叫作马尔可夫性质。用数学表达式表示就是下面的样子：

假设这个模型的每个状态都只依赖于之前的状态，这个假设被称为马尔科夫假设，这个假设可以大大的简化这个问题。显然，这个假设可能是一个非常糟糕的假设，导致很多重要的信息都丢失了。

三要数：

状态：多种特征

初始向量：一开始所处状态的概率

状态转移矩阵：状态互相转移的概率，写成矩阵主要是更方便观察，其中行代表处于第i种状态，列代表转移到第j种状态的概率。

隐马尔可夫模型（HMM）：

在某些情况下，单一的马尔可夫过程并不能很好的描述我们希望发现的模式，我们将不能直观的发现的称之为隐含状态，这样，状态集合就分为两部分，这个算法就叫隐马尔可夫模型。

隐马尔可夫三大问题：

1. 给定模型，如何有效计算产生观测序列的概率？换言之，如何评估模型与观测序列之间的匹配程度？
2. 给定模型和观测序列，如何找到与此观测序列最匹配的状态序列？换言之，如何根据观测序列推断出隐藏的模型状态？
3. 给定观测序列，如何调整模型参数使得该序列出现的概率最大？换言之，如何训练模型使其能最好地描述观测数据？

前两个问题是模式识别的问题：1）根据隐马尔科夫模型得到一个可观察状态序列的概率(评价)；2）找到一个隐藏状态的序列使得这个序列产生一个可观察状态序列的概率最大(解码)。第三个问题就是根据一个可以观察到的状态序列集产生一个隐马尔科夫模型（学习）。

对应的三大问题解法：

1. 向前算法(Forward Algorithm)、向后算法(Backward Algorithm)
2. 维特比算法(Viterbi Algorithm)
3. 鲍姆-韦尔奇算法(Baum-Welch Algorithm) (约等于EM算法)

前向算法就是下一个状态是由前一个决定的，不需要考虑别的影响。

因为我们如果遍历计算路径，那么就要对每一个节点的每一条路径进行计算，计算开销太大，而维特比算法的原理就是，只求出每一个节点的最大的一条路径，最后在进行比较，相对来讲，数量集少了很多。

```
import numpy as np
pi = np.array([.25,.25,.25,.25])
# 状态转移矩阵
A = np.array([
    [0,1,0,0],
    [.4,0,.6,0],
    [0,.4,0,.6],
    [0,0,.5,.5]])
# 观测概率矩阵
B = np.array([
    [.5,.5],
    [.3,.7],
    [.6,.4],
    [.8,.2]])
def viterbi_decode(X):
    T, x = len(X), X[0]
    delta = pi * B[:,x]
    varphi = np.zeros((T, N), dtype = int)
    path = [0] * T
    for i in range(1,T):
        delta = delta.reshape(-1,1)
        tmp = delta * A
        varphi[i, :] = np.argmax(tmp, axis=0)
        delta = np.max(tmp, axis=0) * B[:,X[i]]
    path[-1] = np.argmax(delta)
    for i in range(T-1,0,-1):
        path[i-1] = varphi[i, path[i]]
    return path
X = [1,0,1,0,0]
N = 4
viterbi_decode(X)
```