Viterbi

维特比算法实际是用动态规划解隐马尔科夫模型的预测问题，即用动态规划求概率最大的路径，在 HMM 中，这些路径对应着一个状态序列。根据动态规划的原理，如果最优状态序列在时刻 i 通过结点 t_i，那么这一状态序列从结点 t_i 到终点 t 的部分状态序列，对于从 t_i 到 t 所有可能的部分状态序列来说，必须是最优的。因为如果不是最优的，那么我们就能从 t_i 到 t 寻找一个更好的状态序列以加大获得观察序列的概率。

$$\delta_i(T) = \max_{t_0,\ldots,t_{i-1},t} P(t_0,\ldots,t_{i-1},t,w_1,\ldots,w_{i-1})$$

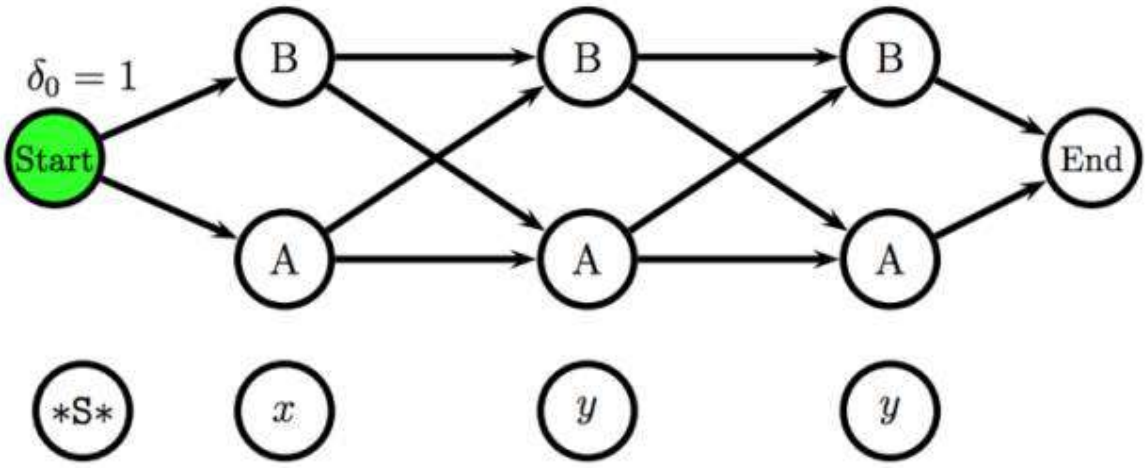
定义在时刻 i 状态为 t 的所有单个路径中概率最大值为 δ，维特比算法可以通过使用马尔科夫假设和如下定义的两个函数计算上式单个路径的最大值。

$$\delta_i(t) = \max_{t_{i-1}} P(t \mid t_{i-1}) \cdot P(w_{i-1} \mid t_{i-1}) \cdot \delta_i(t_{i-1})$$

如下计算每一个状态最可能的前面状态：

$$\Psi_i(t) = \arg\max_{t_{i-1}} P(t \mid t_{i-1}) \cdot P(w \mid t_{i-1}) \cdot \delta_i(t_{i-1})$$

维特比算法使用使用一个被称之为 trellis 的 HMM 表征，它折叠了每一个位置的所有可能状态，并作出了非常明确的独立性假设：每一个位置仅依赖于前一个位置。



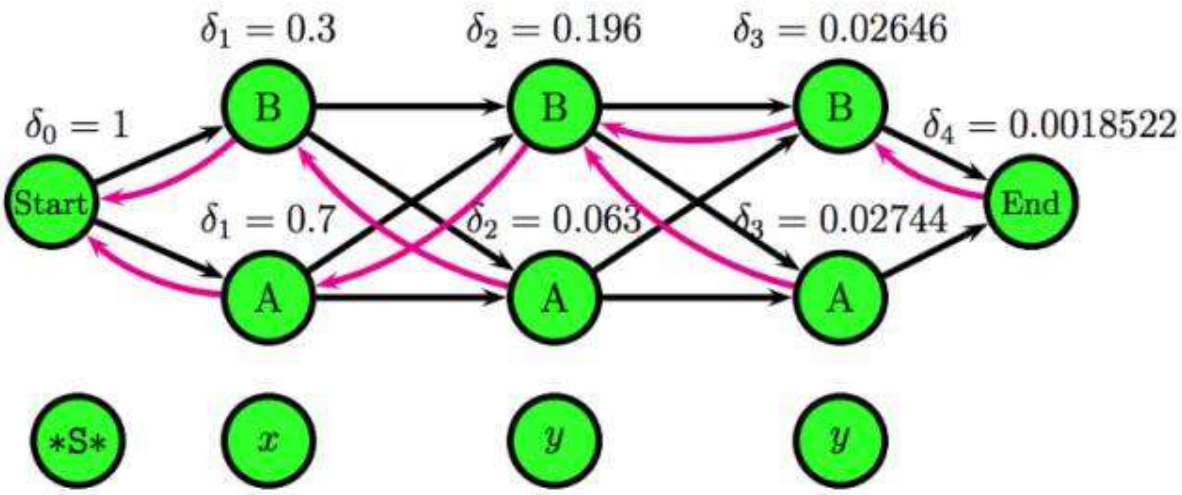
HMM 的 trellis 表示。

	Word		
State	*S*	x	y
Start	1	0	0
A	0	0.4	0.6
B	0	0.3	0.7

	Next		
Current	A	B	End
Start	0.7	0.3	0
A	0.2	0.7	0.1
B	0.7	0.2	0.1

发射概率矩阵和状态迁移概率矩阵。

通过使用维特比算法，转移概率矩阵，我们可以将数据填充到 trellis 图表中，并快速高效地找到维特比路径。



将数据填入 trellis 表示中。

上图是 Roger Levy 展示的维特比算法，完全的案例可参考：http://www.davidsbatista.net/assets/documents/posts/2017-11-12-hmm_viterbi_mini_example.pdf。

维特比算法总结

（1）从点S出发，对于第一个状态X1的各个节点，不妨假定有n1个，计算出S到它们的距离

d(S,X1i)，其中X1i代表任意状态1的节点。因为只有一步，所以这些距离都是S到它们各自的最短距离。

（2）对于第二个状态X2的所有节点，要计算出从S到它们的最短距离。对于特点的节点X2i，从S到它的路径可以经过状态1的n1中任何一个节点X1i，对应的路径长度就是d(S,X2i) = d(S,X1i) + d(X1i,X2i)。

由于j有n1种可能性，我们要一一计算，找出最小值。即：d(S,X2i) = minI=1，n1 d(S,X1i) + d(X1i,X2i)

这样对于第二个状态的每个节点，需要n1次乘法计算。假定这个状态有n2个节点，把S这些节点的距离都算一遍，就有O(n1·n2)次计算。

（3）接下来，类似地按照上述方法从第二个状态走到第三个状态，一直走到最后一个状态，就得到了整个网格从头到尾的最短路径。每一步计算的复杂度都和相邻两个状态Si和Si+1各自的节点数目ni，ni+1的乘积成正比，即O(ni·ni+1)

（4）假设这个隐含马尔可夫链中节点最多的状态有D个节点，也就是说整个网格的宽度为D，那么任何一步的复杂度不超过O(D2)，由于网格长度是N，所以整个维特比算法的复杂度是O(N·D2)。

算法(维特比算法)

输入：模型$\boldsymbol{\lambda}=(\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})$和观测$O=(o_1,o_2,\cdots,o_T)$;

输出：最优路径$I^*=(i_1^*,i_2^*,\cdots,i_T^*)$。

⑴初始化

$$\delta_1(i)=\pi_i b_i(o_1)\,,\qquad i=1,2,\cdots,N$$
$$\psi_1(i)=0\,,\qquad i=1,2,\cdots,N$$

(2)递推。对$t=2,3,\cdots,T$

$$\delta_t(i)=\max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}]b_i(o_t)\,,\qquad i=1,2,\cdots,N$$
$$\psi_t(i)=\arg\max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}]\,,\qquad i=1,2,\cdots,N$$

(3)终止

$$P^*=\max_{1\leqslant i\leqslant N}\delta_T(i)$$
$$i_T^*=\arg\max_{1\leqslant i\leqslant N}[\delta_T(i)]$$

(4)最优路径回溯。对$t=T-1,T-2,\cdots,1$

$$i_t^*=\psi_{t+1}(i_{t+1}^*)$$

求得最优路径$I^*=(i_1^*,i_2^*,\cdots,i_T^*)$。

```python
# We implement the viterbi decoding algorithm.
def viterbi(words, hmm):
    '''
    Viterbi algorihtm.

    Parameters
    ----------
    words: list(str)
        The list of words
    hmm: HMM
        The hmm model

    Return
    ------
    result: list(str)
        The POS-tag for each word.
    '''
    # unpack the length of words, and number of postags
    N, T = len(words), len(hmm.postags)

    # allocate the decode matrix
    score = [[-float('inf') for j in range(T)] for i in range(N)]
    path = [[-1 for j in range(T)] for i in range(N)]

    for i, word in enumerate(words):
        if i == 0:
            for j, tag in enumerate(hmm.postags):
                score[i][j] = hmm.emit(words, i, tag)
        else:
            for j, tag in enumerate(hmm.postags):
                best, best_t = -1e20, -1
                temp = 0
```