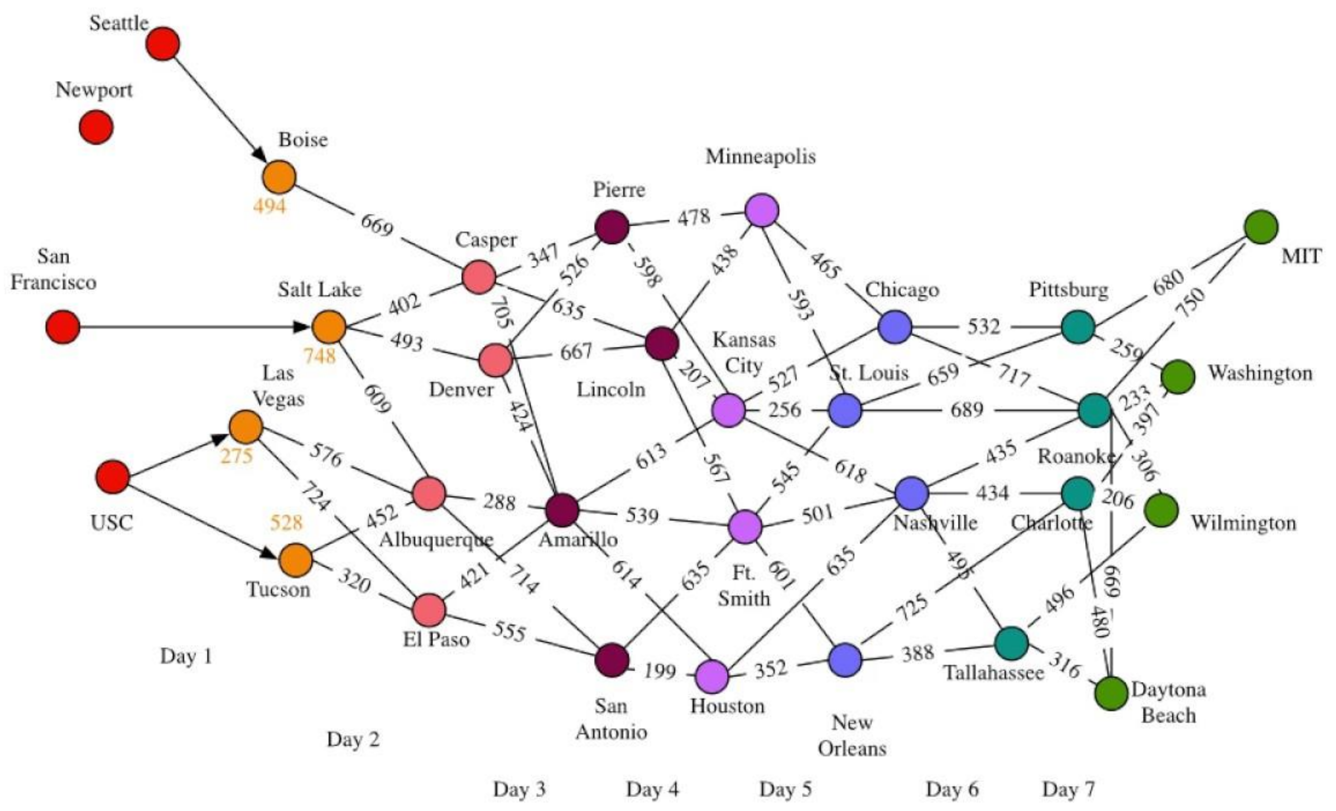
Viterbi algorithm



Name	محمد علاء فرج خميس
ID	18011559

Definition

The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimates of the most likely sequence of hidden states—called the Viterbi path—those results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).

The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, speech synthesis, diarization keyword spotting, computational linguistics, and bioinformatics. For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal.

History

The Viterbi algorithm is named after Andrew Viterbi, who proposed it in 1967 as a decoding algorithm for convolutional codes over noisy digital communication links. It has, however, a history of multiple invention, with at least seven independent discoveries, including those by Viterbi, Needleman and Wunsch, and Wagner and Fischer. It was introduced to Natural Language Processing as a method of part-of-speech tagging as early as 1987.

Viterbi path and Viterbi algorithm have become standard terms for the application of dynamic programming algorithms to maximization problems involving probabilities. For example, in statistical parsing a dynamic programming algorithm can be used to discover the single most likely context-free derivation (parse) of a string, which is commonly called the "Viterbi parse". Another application is in target tracking, where the track is computed that assigns a maximum likelihood to a sequence of observations.

Extensions

A generalization of the Viterbi algorithm, termed the max-sum algorithm (or max-product algorithm) can be used to find the most likely assignment of all or some subset of latent variables in a large number of graphical models, e.g. Bayesian networks, Markov random fields and conditional random fields. The latent variables need, in general, to be connected in a way somewhat similar to an hidden Markov model (HMM), with a limited number of connections between variables and some type of linear structure among the variables. The general algorithm involves message passing and is substantially similar to the belief propagation algorithm (which is the generalization of the forward-backward algorithm).

With the algorithm called iterative Viterbi decoding one can find the subsequence of an observation that matches best (on average) to a given hidden Markov model. This algorithm is proposed by Qi Wang et al. to deal with turbo code. Iterative Viterbi decoding works by iteratively invoking a modified Viterbi algorithm, resituating the score for a filler until convergence.

An alternative algorithm, the Lazy Viterbi algorithm, has been proposed. For many applications of practical interest, under reasonable noise conditions, the lazy decoder (using Lazy Viterbi algorithm) is much faster than the original Viterbi decoder (using Viterbi algorithm). While the original Viterbi algorithm calculates every node in the trellis of possible outcomes, the Lazy Viterbi algorithm maintains a prioritized list of nodes to evaluate in order, and the number of calculations required is typically fewer (and never more) than the ordinary Viterbi algorithm for the same result. However, it is not so easy to

Example for the Viterbi algorithm

Markov Model inference with the Viterbi algorithm

In this mini-example, we'll cover the problem of inferring the most-likely state sequence given an HMM and an observation sequence. The problem of parameter estimation is not covered.

Once again, the dynamic program for the HMM trellis on an observation sequence of length n is as follows:

1. Initialize $\delta_0(s) = 1$ for s the start state, and $\delta_0(s) = 0$ for all other states (this is equivalent to having only the start state in the trellis at position zero)
2. For each value $i = 1, \ldots, n$, calculate:
   (a) $\delta_i(s) = \max_{s_{i-1}} P(s_i|s_{i-1})P(w_{i-1}|s_{i-1})\delta_{i-1}(s_{i-1})$
   (b) $\psi_i(s) = \arg\max_{s_{i-1}} P(s_i|s_{i-1})P(w_{i-1}|s_{i-1})\delta_{i-1}(s_{i-1})$
3. Finally, fill out the end state of the trellis (position $n+1$) using the rules in (2) above.

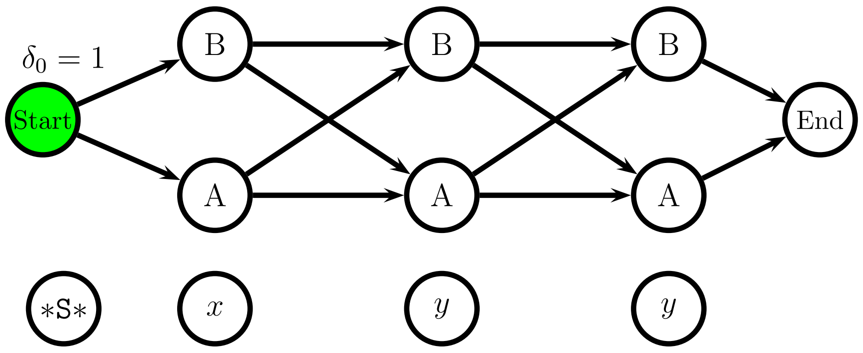
We'll take as our transition probability distribution

	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

and as our emission probability distribution

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

Suppose we see the input sequence $x\ y\ y$. We start by constructing the trellis and initializing it with δ_0(*START*) $= 1$ at the start. The green nodes indicate how much of the sequence can be considered generated after each iteration of trellis-filling:



Next, we calculate the δ values at position 1:

$$\delta_1(A) = \max_{s_0} P(A|s_0)P(*\texttt{S}*|s_0)\delta_0(s_0) \tag{1}$$

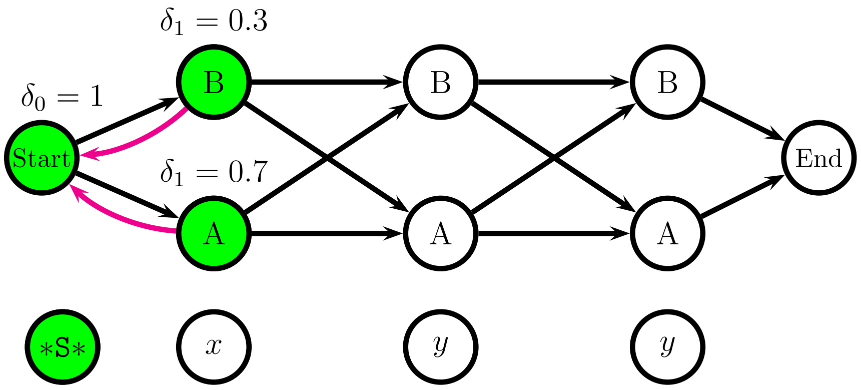
which is simple since there is only one possible value s_0, the start state:

$$\delta_1(A) = 1 \times 1 \times 0.7 \tag{2}$$
$$= 0.7 \tag{3}$$

Likewise, we obtain

$$\delta_1(B) = 1 \times 1 \times 0.3 \tag{4}$$

The backtraces are both trivial as well: $\psi_1(A) = \psi_1(B) = *\texttt{S}*_0$



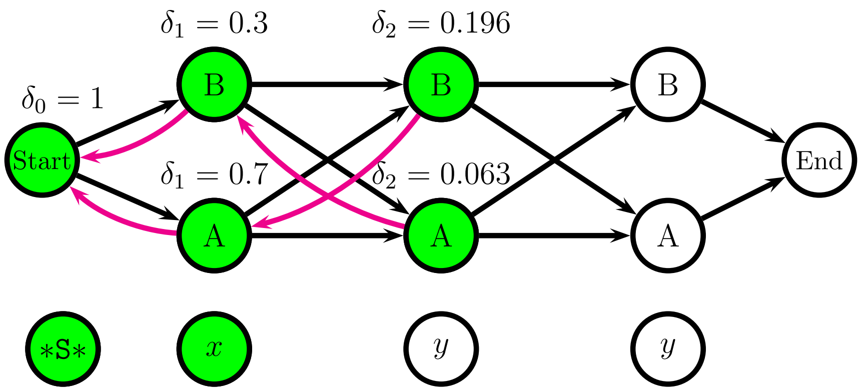
We next calculate the δ values at position 2:

$$\delta_2(A) = \max_{s_1} P(A|s_1)P(*\texttt{S}*|s_1)\delta_1(s_1) \tag{5}$$
$$= \max\{0.2 \times 0.4 \times 0.7, 0.7 \times 0.3 \times 0.3\} \tag{6}$$
$$= \max\{0.056, 0.063\} \tag{7}$$
$$= 0.063 \tag{8}$$

This value was higher for $s_1 = B$, hence $\psi_2(A) = B_1$. We also have

$$\delta_2(B) = \max\{\overbrace{0.7 \times 0.4 \times 0.7}^{A}, \overbrace{0.2 \times 0.3 \times 0.3}^{B}\} \tag{9}$$

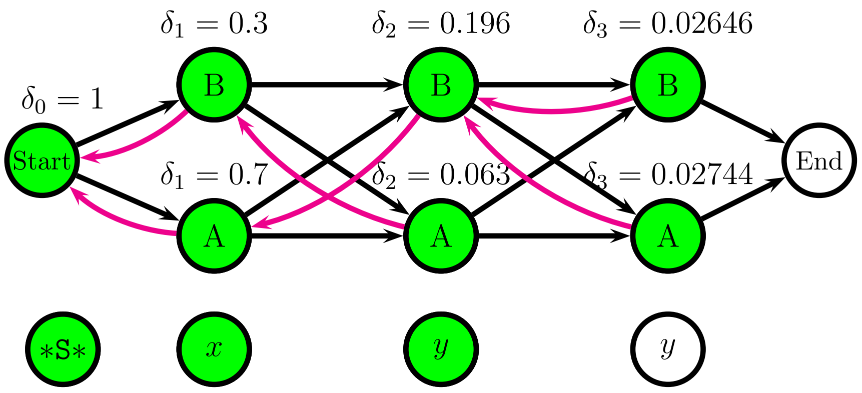
giving us $\psi_2(B) = A_1$:



We recurse one more time for position 3:

$$\delta_3(A) = \max\{\overbrace{0.2 \times 0.6 \times 0.063}^{A}, \overbrace{0.7 * 0.7 * 0.196}^{B}\} \tag{10}$$

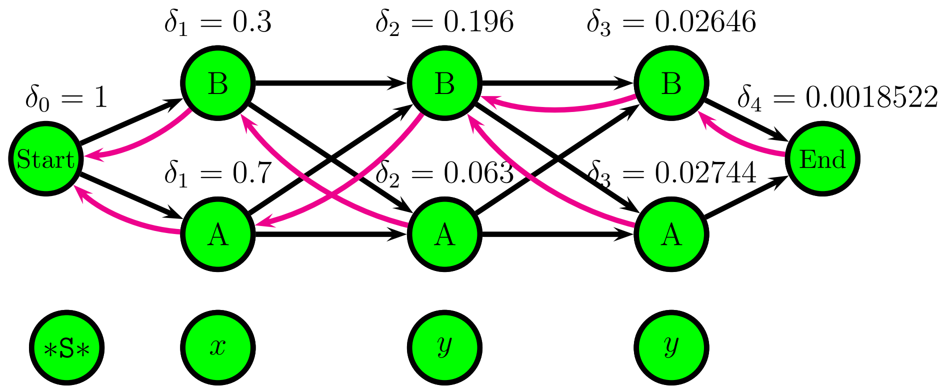
$$\delta_3(B) = \max\{\overbrace{0.7 \times 0.6 \times 0.063}^{A}, \overbrace{0.2 * 0.7 * 0.196}^{B}\} \tag{11}$$



and finally the last time for the end state:

$$\delta_4(End) = \max\{\overbrace{0.1 \times 0.6 \times 0.02744}^{A}, \overbrace{0.1 * 0.7 * 0.02646}^{B}\} \tag{12}$$

giving us



We made it! From the end state we can read off the Viterbi sequence (following the backtraces through to the start state) and its probability:

Viterbi sequence: ABB
$P(ABB, xyy) = 0.00185522$

Note that this is different than the inference than would be made either with a "reverse emission model" where $P(A|x) = 0.4, P(A|y) = 0.3$, which would favor the sequence BBB, or with the transition model alone (which would favor the sequence ABA). The emission and transition models work together to determine the posterior inference.