CS378 Lecture Note: Viterbi Algorithm

1 Viterbi Algorithm

The Viterbi algorithm is an algorithm for performing inference in Hidden Markov Models. Briefly, a Hidden Markov Model is defined by

$$P(\mathbf{y}, \mathbf{x}) = P_S(y_1) P_E(x_1|y_1) \left[\prod_{i=2}^{n} P_T(y_i|y_{i-1}) P_E(x_i|y_i) \right] P_T(\text{STOP}|y_n) \tag{1}$$

(there are many correct ways to write this formula). We want to compute $\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$, the most likely tag sequence given some input words $\mathbf{x}$. This is equivalent to computing $\arg\max_{\mathbf{y}} \frac{P(\mathbf{y},\mathbf{x})}{P(\mathbf{x})} = \arg\max_{\mathbf{y}} P(\mathbf{y}, \mathbf{x})$, due to Bayes' rule and the fact that the ensuing denominator does not depend on $\mathbf{y}$.

Let $|T|$ be the number of tags and let n be the length of a sentence under consideration. Assume that tags and words are both *indexed*, so that both words and tags can be represented as integers. The set of tags should contain a STOP token.

Model Viterbi requires the model parameters as input, which are typically estimated from training data.

1. Initial (start) probabilities: $\log P_S(y_1 = y)$. These can be stored in a vector $S[i] = \log P_S(y = i)$
2. Transition probabilities: $\log P_T(y_i = y|y_{i-1} = y_{\text{prev}})$. These can be stored in a matrix $T[i, j] = \log P_T(y = j|y_{\text{prev}} = i)$
3. Emission probabilities: $\log P_E(x_i = x|y = y_i)$. These can be stored in a matrix $E[i, j] = \log P_E(x = j|y = i)$

Algorithm We're now ready to describe the algorithm (see next page):

Algorithm 1 Viterbi Algorithm

function VITERBI(x, S, T, E) ▷ x: sentence of length n, S: initial log probs, T: transition log probs, E: emission log probs
  Initialize v, a $n \times |T|$ matrix
  for $y = 1$ to $|T|$ **do** ▷ Handle the initial state
    $v[1, y] = S[y] + E[y, x_1]$
  end for
  for $i = 2$ to n **do**
    for $y = 1$ to $|T|$ **do**
      $v[i, y] = E[y, x_i] + \max_{y_{\text{prev}}} \left(T[y_{\text{prev}}, y] + v[i-1, y_{\text{prev}}] \right)$
    end for
  end for
  for $y = 1$ to $|T|$ **do** ▷ Handle the final state
    $v[n, y] = v[n, y] + T[y, \text{STOP}]$
  end for
  Best final state = $\arg\max_y v[n, y]$
  By tracking argmaxes in the algorithm in addition to maxes, you can reconstruct the answer
end function

For further reference, Wikipedia has a similar implementation,[1] but not in log space. That is, rather than adding up log probabilities, they multiply probabilities, which is riskier numerically.

1.1 Variants

The critical operations in the above algorithm are the + operations (to combine log probability values) and the max operations

1. $\max$, +: Viterbi algorithm in log space, as shown above (expects log-probability matrices as input)

2. $\max$, $\times$: Viterbi algorithm in real space (expects probability matrices as input)

3. +, $\times$: sum-product algorithm (also called the forward algorithm) in real space. Can be used to compute $P(\mathbf{x}) = \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y})$. Can be combined with a version of this algorithm called the backward algorithm to compute $P(y_i|\mathbf{x})$ for each position i in the sentence. This is outside the scope of this course, but is discussed more in the textbook.

4. log-sum, +: sum-product algorithm in log space. log-sum$(a, b) = \log(\exp(a) + \exp(b))$; that is, it takes two log-probabilities, turns them into probabilities, adds them together, and re-logs them. There is no other way to "add" log probabilities, since adding in log space means multiplying the underlying probabilities.

[1] https://en.wikipedia.org/wiki/Viterbi_algorithm