9: Viterbi Algorithm for HMM Decoding
Machine Learning and Real-world Data

Andreas Vlachos
(adapted from Simone Teufel)

Dept. of Computer Science and Technology
University of Cambridge

Lent 2020
The dishonest casino, dice edition.

Two hidden states: \( L \) (loaded dice), \( F \) (fair dice).

Input: dual tape of state and observation (dice outcome) sequences \( X \) and \( O \).

You estimated transition and emission probabilities (\( A \) and \( B \), Task 7).
This session: decoding

- Now we can only observe the numbers that are thrown but we don’t know which dice is currently in use. (more realistic unless the croupier is a friend?)
- We want the HMM to find out when the fair dice was out, and when the loaded dice was out.
- We need to write a decoder. (Task 8)
Decoding: finding the most likely path

Definition of decoding: Finding the most likely hidden state sequence $X$ that explains the observation $O$ given the HMM parameters $\mu = (A, B)$.

\[
\hat{X} = \arg\max_X P(X, O; \mu)
\]

\[
= \arg\max_X P(O|X; B)P(X; A)
\]

\[
= \arg\max_{X_1 \ldots X_T} \prod_{t=1}^{T} P(O_t|X_t; B)P(X_t|X_{t-1}; A)
\]

Number of possible state sequences $X$ is $O(N^T)$ ($N =$ number of unique hidden states); too large for brute force search.
(Reminder from Algorithms course)
We can use Dynamic Programming if two conditions apply:

- **Optimal substructure property**
  - An optimal state sequence $X_1 \ldots X_j \ldots X_T$ contains inside it the sequence $X_1 \ldots X_j$, which is also optimal

- **Overlapping subsolutions property**
  - If both $X_t$ and $X_u$ are on the optimal path, with $u > t$, then the calculation of the probability for being in state $X_t$ is part of each of the many calculations for being in state $X_u$. 
The intuition behind Viterbi

- Here’s how we can save ourselves a lot of time.
- Because of the Limited Horizon of the HMM, we don’t need to keep a complete record of how we arrived at a certain state.
  - For 1st-order HMM, we need to record one previous step.
- Just do the calculation of the probability of reaching each state once for each time step and memoise it in an appropriate data structure
  - This reduces our effort to $O(N^2T)$ for the 1st order HMM.
  - We need to calculate the probability of arriving in each hidden state given each previous hidden state for every timestep.
- What if we had a 2nd order HMM?
Viterbi: main data structure

- Memoisation is done using a *trellis*.
- A trellis is equivalent to a Dynamic Programming table.
- The trellis is \((N + 2) \times (T + 2)\) in size, with states \(j\) as rows and time steps \(t\) as columns.
- Each cell \(j, t\) records the Viterbi probability \(\delta_j(t)\), the probability of the most likely path that ends in state \(s_j\) at time \(t\):
  \[
  \delta_j(t) = \max_{1 \leq i \leq N} [\delta_i(t - 1) a_{ij} b_j(O_t)]
  \]
- This probability is calculated by maximising over the best ways of going to \(s_j\) for each \(s_i\).
- \(a_{ij}\): the transition probability from \(s_i\) to \(s_j\)
- \(b_j(O_t)\): the probability of emitting \(O_t\) from destination state \(s_j\)
Viterbi algorithm, initialisation

Note: the probability of a state starting the sequence at $t = 0$ is just the probability of it emitting the first symbol.
Viterbi algorithm, initialisation
Viterbi algorithm, initialisation
Viterbi algorithm, initialisation

\[ b_0(k_0) \]

\[ \delta_0(0) = b_0(k_0) \]
Viterbi algorithm, main step

\[ \delta_0(0) \]

\[ S_0 \rightarrow S_F \]
\[ S_0 \rightarrow S_L \]

\[ X_1 = F \]
\[ X = L \]
Viterbi algorithm, main step: observation is 4

\[ o_1 = 4 \]

\[ \delta_0(0) \]

\[ S_0 \rightarrow S_F \]

\[ b_{F}(4) \]

\[ a_{0F} \]

\[ a_{0L} \]

\[ X_1 = F \]

\[ X = L \]

\[ X_1 = L \]
Viterbi algorithm, main step: observation is 4

\[
\delta_F(1) = a_{OF}b_F(4)\delta_0(0)
\]

\[
o_1 = 4
\]

\[
x_1 = F
\]

\[
x = L
\]
Viterbi algorithm, main step, $\psi$

- $\psi_j(t)$ is a helper variable that stores the $t - 1$ state index $i$ on the highest probability path.

$$\psi_j(t) = \arg\max_{1 \leq i \leq N} [\delta_i(t - 1) a_{ij} b_j(O_t)]$$

- In the backtracing phase, we will use $\psi$ to find the previous cell/state in the best path.
Viterbi algorithm, main step: observation is 4

\[ o_1 = 4 \]

\[ \delta_F(1) = a_{0F} b_F(4) \delta_0(0) \]

\[ X_1 = F \]

\[ X = L \]

\[ X = L \]
Viterbi algorithm, main step: observation is 4

\[
\delta_F(1) = a_{OF} b_F(4) \delta_0(0) \\
\psi_F(1) = 0
\]

\[X_1 = F \quad X = L \quad X = L\]
Viterbi algorithm, main step: observation is 4

\[ o_1 = 4 \]

\[ \delta_F(1) = a_{0F} b_F(4) \delta_0(0) \]

\[ \varphi_F(1) = 0 \]

\[ X_1 = F \]

\[ X = L \]
Viterbi algorithm, main step: observation is 4

\[ o_1 = 4 \]

\[ \delta_F(1) = a_{0F} b_F(4) \delta_0(0) \]

\[ \psi_F(1) = 0 \]

\[ \delta_L(1) = a_{0L} b_L(4) \delta_0(0) \]

\[ \psi_L(1) = 0 \]

\[ X_1 = F \]

\[ X = L \]

\[ X = L \]
Viterbi algorithm, main step: observation is 3

\[ o_1 = 4 \quad o_2 = 3 \]

\[ \delta_F(1) \quad \delta_L(1) \]

\[ S_0 \xrightarrow{S_F} \quad S_F \xrightarrow{S_F} \quad S_L \xrightarrow{S_L} \]

\[ a_{FF} \quad a_{LF} \quad b_{F(3)} \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]
Viterbi algorithm, main step: observation is 3

\[ o_1 = 4 \quad o_2 = 3 \]

\[ \delta_F(1) \quad \delta_L(1) \]

\[ S_F \quad S_L \]

\[ X_1 = F \quad X_2 = L \quad X = L \]
Viterbi algorithm, main step: observation is 3

\[ \delta_F(2) = \max (a_{FF} b_F(3) \delta_F(1), a_{LF} b_F(3) \delta_L(1)) \]

\[ \psi_F(2) = L \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]
Viterbi algorithm, main step: observation is 3

$\delta_F(1) \rightarrow S_F \quad \delta_L(1) \rightarrow S_L$

$\delta_F(2) \rightarrow S_F \quad \delta_L(2) = \max(a_{FL} b_L(3) \delta_F(1), a_{LL} b_L(3) \delta_L(1))$

$X_1 = F \quad X_2 = L \quad X_3 = L$

$o_1 = 4 \quad o_2 = 3$
Viterbi algorithm, main step: observation is 3

\[ o_1 = 4 \quad o_2 = 3 \]

\[ S_0 \rightarrow S_F \quad \delta_F(1) \quad a_{FL} \quad S_F \rightarrow S_L \quad \delta_L(1) \]

\[ S_L \rightarrow S_F \quad \delta_F(2) \quad b_L(3) \quad S_F \rightarrow S_L \quad \delta_L(2) = \max \left( a_{FL} b_L(3) \delta_F(1), \quad a_{LL} b_L(3) \delta_L(1) \right) \]

\[ \psi_L(2) = F \]

\[ X_1 = F \quad X_2 = L \quad X = L \]
Viterbi algorithm, main step: observation is 5

\[ \delta_F(3) = \max (a_{FF} b_F(5) \delta_F(2), a_{LF} b_F(5) \delta_L(2)) \]

\[ \psi_F(3) = F \]

\[ X_1 = F \]
\[ X_2 = L \]
\[ X_3 = L \]
Viterbi algorithm, main step: observation is 5

\[ \delta_F(2) \]

\[ \delta_L(2) \]

\[ \delta_L(3) = \max (a_{FL} b_L(5) \delta_F(2), a_{LL} b_L(5) \delta_L(2)) \]

\[ \psi_L(3) = L \]

\[ X_1 = F \]

\[ X_2 = L \]

\[ X_3 = L \]

\[ o_1 = 4 \]

\[ o_2 = 3 \]

\[ o_3 = 5 \]
Viterbi algorithm, termination

\[ \delta_f(4) = \max (a_{fF} b_{f(k)}, \delta_F(3), a_{LF} b_{L(k)} \delta_L(3)) \]
Viterbi algorithm, termination

\[ \delta_F(3) \]

\[ \delta_L(3) \]

\[ \delta_f(4) = \max(a_F b(4) \delta_F(3), a_L b(4) \delta_L(3)) \]

\[ \psi_f(4) = L \]
Viterbi algorithm, backtracing

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]

\[ \psi_f(4) = L \]
Viterbi algorithm, backtracing

$V_1 = 4$  $V_2 = 3$  $V_3 = 5$

$X_1 = F$  $X_2 = L$  $X_3 = L$

$V_f(4) = L$
Viterbi algorithm, backtracing

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[ S_0 \quad S_{F} \quad S_{L} \quad S_{F} \quad S_{F} \quad S_{F} \quad S_{L} \quad S_{L} \quad S_{L} \quad S_f \]

\[ \psi_L(3) = L \]

\[ X_1 = \quad X_2 = L \quad X_3 = L \]
Viterbi algorithm, backtracking

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]
Viterbi algorithm, backtracing

V_\psi_2 = F

X_1 = F
X_2 = L
X_3 = L
Viterbi algorithm, backtracing

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[ \psi_L(2) = F \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]
Viterbi algorithm, backtracing

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[
\begin{align*}
S_0 & \rightarrow S_F & S_L \rightarrow S_F & \rightarrow S_F & \rightarrow S_F & \rightarrow S_f \\
S_L & \rightarrow S_F & S_F \rightarrow S_L & \rightarrow S_L & \rightarrow S_f \\
X_1 & = F & X_2 & = L & X_3 & = L
\end{align*}
\]
Viterbi algorithm, backtracing

\[ o_1 = 4 \quad o_2 = 3 \quad o_3 = 5 \]

\[ X_1 = F \quad X_2 = L \quad X_3 = L \]
Why is it necessary to keep $N$ states at each time step?

- We have convinced ourselves that it’s not necessary to keep more than $N$ (“real”) states per time step.
- But could we cut down the table to just a one-dimensional table of $T$ time slots by choosing the probability of the best path overall ending in that time slot, in any of the states?
  - This would be the greedy choice
  - But think about what could happen in a later time slot.
  - You could encounter a zero or very low probability concerning all paths going through your chosen state $s_j$ at time $t$.
  - Now a state $s_k$ that looked suboptimal in comparison to $s_j$ at time $t$ becomes the best candidate.
  - As we don’t know the future, we need to keep the probabilities for each state at each timestep.
- Viterbi is an exact decoding algorithm, find the same solution as brute-force but faster!
Precision and Recall

- So far, we have measured system success in accuracy.
- But sometimes it’s only one type of instances that we find interesting.
- We don’t want a summary measure that averages over interesting and non-interesting instances, as accuracy does.
- In those cases, we use precision, recall and F-measure.
- These metrics are imported from the field of information retrieval, where the difference between interesting and non-interesting examples is particularly high.
- Accuracy doesn’t work well when the types of instances are unbalanced.
Precision and Recall

System says:

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>a</td>
<td>b</td>
<td>a+b</td>
</tr>
<tr>
<td>L</td>
<td>c</td>
<td>d</td>
<td>c+d</td>
</tr>
<tr>
<td>Total</td>
<td>a+c</td>
<td>b+d</td>
<td>a+b+c+d</td>
</tr>
</tbody>
</table>

Truth is:

- Precision of L: \( P_L = \frac{d}{b+d} \)
- Recall of L: \( R_L = \frac{d}{c+d} \)
- F-measure of L: \( F_L = \frac{2P_L R_L}{P_L + R_L} \)
- Accuracy: \( A = \frac{a+d}{a+b+c+d} \)
Your task today

Task 8:

- Implement the Viterbi algorithm.
- Run it on the dice dataset and measure precision of $L (P_L)$, recall of $L (R_L)$ and F-measure of $L (F_L)$.
Literature

- Smith, Noah A. (2011). Linguistic Structure Prediction (section 3.3.3)
- Extra reading on the connection between Viterbi and Dijkstra’s algorithms (likely sequence vs shortest path): Liang Huang’s tutorial: https://www.aclweb.org/anthology/C08-5001.pdf