Introduction to Natural Language Processing I
[Statistické metody zpracování přirozených jazyků I]
(NPFL067)
http://ufal.mff.cuni.cz/courses/npfl067

prof. RNDr. Jan Hajič, Dr. / doc. RNDr. Pavel Pecina, Ph.D.
ÚFAL MFF UK

{hajic,pecina}@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/jan-hajic
HMM Algorithms: Trellis and Viterbi
HMM: The Two Tasks

- **HMM (the general case):**
  - five-tuple \((S, S_0, Y, P_S, P_Y)\), where:
    - \(S = \{s_1, s_2, \ldots, s_T\}\) is the set of states, \(S_0\) is the initial state,
    - \(Y = \{y_1, y_2, \ldots, y_V\}\) is the output alphabet,
    - \(P_S(s_j|s_i)\) is the set of prob. distributions of transitions,
    - \(P_Y(y_k|s_i, s_j)\) is the set of output (emission) probability distributions.

- **Given an HMM & an output sequence \(Y = \{y_1, y_2, \ldots, y_k\}\):**
  - (Task 1) compute the probability of \(Y\);
  - (Task 2) compute the most likely sequence of states which has generated \(Y\).
Trellis - Deterministic Output

HMM:

- trellis state: (HMM state, position)
- each state: holds one number (prob): $\alpha$
- probability or Y: $\sum \alpha$ in the last state

\[
p(\text{toe}) = 0.6 \times 0.88 \times 1 + 0.4 \times 1 \times 1 = 0.568
\]

Trellis:

- "rollout"
Creating the Trellis: The Start

- **Start in the start state (\(\times\)),**
  - set its \(\alpha(\times,0)\) to 1.

- **Create the first stage:**
  - get the first “output” symbol \(y_1\)
  - create the first stage (column)
  - but only those trellis states which generate \(y_1\)
  - set their \(\alpha(state,1)\) to \(P_S(state|\times) \alpha(\times,0)\)

- ...and forget about the 0-th stage
Trellis: The Next Step

• Suppose we are in stage $i$

• Creating the next stage:
  – create all trellis states in the next stage which generate $y_{i+1}$, but only those reachable from any of the stage-$i$ states
  – set their $\alpha(state, i+1)$ to:
    \[ P_S(state|prev.state) \times \alpha(prev.state, i) \]
    (add up all such numbers on arcs going to a common trellis state)
  – ...and forget about stage $i$
Trellis: The Last Step

- Continue until “output” exhausted
  - $|Y| = 3$: until stage 3
- Add together all the $\alpha(state, |Y|)$
- That’s the $P(Y)$.  
- Observation (pleasant):
  - memory usage max: $2|S|$
  - multiplications max: $|S|^2|Y|$

\[ \alpha = 0.568 \]

\[ P(Y) = 0.568 \]
Trellis: The General Case (still, bigrams)

- Start as usual:
  - start state (′), set its $\alpha(′,0)$ to 1.

\[
p(\text{toe}) = 0.48 \cdot 0.616 \cdot 0.6 + \\
0.2 \cdot 0.1 \cdot 0.176 + \\
0.2 \cdot 0.1 \cdot 0.12 \approx 0.237
\]
General Trellis: The Next Step

• We are in stage $i$:
  – Generate the next stage $i+1$ as before (except now arcs generate output, thus use only those arcs marked by the output symbol $y_{i+1}$)
  – For each generated state, compute $\alpha(state, i+1) = \sum_{\text{incoming arcs}} P_Y(y_{i+1}|state, \text{prev.state}) \times \alpha(\text{prev.state}, i)$

...and forget about stage $i$ as usual.
Trellis: The Complete Example

Stage:

\[ \begin{array}{c|c|c|c}
   & 0 \rightarrow 1 & 1 \rightarrow 2 & 2 \rightarrow 3 \\
   y_1: t & A_0 \alpha = .48 & A_1 \alpha = .2 & A_2 \alpha = .024 + .177408 = .201408 \\
   & \alpha = 1 & \ & \alpha = .29568 \\
   y_2: o & C_0 \alpha = .2 & C_1 & B_3 \alpha = .024 + .177408 = .201408 \\
   & \ & \ & + \\
   y_3: e & D_0 \alpha = .06 & \ & = .035200 \\
   & \ & \ & \downarrow \\
   & \ & \ & P(Y) = P(\text{toe}) = .236608 \\
\end{array} \]
The Case of Trigrams

- Like before, but:
  - states correspond to bigrams,
  - output function always emits the second output symbol of the pair (state) to which the arc goes:

```
  p(toe) = .6 .88 .07 \geq .037
```

Multiple paths not possible \(\rightarrow\) trellis not really needed
Trigrams with Classes

- More interesting:
  - n-gram class LM: \( p(w_i|w_{i-2},w_{i-1}) = p(w_i|c_i) \ p(c_i|c_{i-2},c_{i-1}) \)
  - states are pairs of classes \((c_{i-1},c_i)\), and emit "words":

  (letters in our example)

  \[
  \begin{align*}
  p(t|C) &= 1 & \text{usual,} \\
  p(o|V) &= .3 & \text{non-} \\
  p(e|V) &= .6 & \text{overlapping} \\
  p(y|V) &= .1 & \text{classes} \\
  p(\text{toe}) &= .6 \ 1 \ .88 \ .3 \ .07 \ .6 \approx .00665 \\
  p(\text{teo}) &= .6 \ 1 \ .88 \ .6 \ .07 \ .3 \approx .00665 \\
  p(\text{toy}) &= .6 \ 1 \ .88 \ .3 \ .07 \ .1 \approx .00111 \\
  p(\text{tty}) &= .6 \ 1 \ .12 \ 1 \ 1 \ .1 \approx .0072
  \end{align*}
  \]
Class Trigrams: the Trellis

• Trellis generation (Y = “toy”):

\[
\begin{align*}
p(t|C) &= 1 \\
p(o|V) &= .3 \\
p(e|V) &= .6 \\
p(y|V) &= .1
\end{align*}
\]

\[
\begin{align*}
\alpha &= 1 \\
\alpha &= .6 \times .1 \\
\alpha &= .1584 \times .07 \times .1 \\ &\approx .00111
\end{align*}
\]

Again, trellis useful but not really needed.
Overlapping Classes

• Imagine that classes may overlap
  – e.g. ‘r’ is sometimes vowel sometimes consonant, belongs to V as well as C:

\[
\begin{align*}
p(t|C) &= .3 \\
p(r|C) &= .7 \\
p(o|V) &= .1 \\
p(e|V) &= .3 \\
p(y|V) &= .4 \\
p(r|V) &= .2
\end{align*}
\]

\[p(\text{try}) = ?\]
Overlapping Classes: Trellis Example

\[ p(t|C) = .3 \]
\[ p(r|C) = .7 \]
\[ p(o|V) = .1 \]
\[ p(e|V) = .3 \]
\[ p(y|V) = .4 \]
\[ p(r|V) = .2 \]

\[ \alpha = 1 \]
\[ \alpha = .18 \times .12 \times .7 = .01512 \]
\[ \alpha = .03168 \times .07 \times .4 \approx .0008870 \]

\[ p(Y) = .006935 \]
Trellis: Remarks

• So far, we went left to right (computing $\alpha$)
• Same result: going right to left (computing $\beta$)
  – supposed we know where to start (finite data)
• In fact, we might start in the middle going left and right
• Important for parameter estimation
  (Forward-Backward Algorithm alias Baum-Welch)
• Implementation issues:
  – scaling/normalizing probabilities, to avoid too small numbers & addition problems with many transitions
The Viterbi Algorithm

• Solving the task of finding the most likely sequence of states which generated the observed data
• i.e., finding

\[ S_{\text{best}} = \arg\max_S P(S|Y) \]

which is equal to (Y is constant and thus P(Y) is fixed):

\[ S_{\text{best}} = \arg\max_S P(S,Y) = \]

\[ = \arg\max_S P(s_0,s_1,s_2,\ldots,s_k,y_1,y_2,\ldots,y_k) = \]

\[ = \arg\max_S \prod_{i=1..k} p(y_i|s_i,s_{i-1})p(s_i|s_{i-1}) \]
The Crucial Observation

- Imagine the trellis build as before (but do not compute the $\alpha$'s yet; assume they are o.k.); stage $i$:

  \[
  \begin{array}{c}
  \text{stage} \\
  1 & 2 \\
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{stage} \\
  1 & 2 \\
  \end{array}
  \]

NB: remember previous state from which we got the maximum:

\[
\begin{array}{c}
\alpha = .6 \\
\alpha = .4 \\
\alpha = \max(.3,.32) = .32 \\
\end{array}
\]

this is certainly the “backwards” maximum to (D,2)... but it cannot change even whenever we go forward (M. Property: Limited History)
Viterbi Example

- ‘r’ classification (C or V?, sequence?):

```
V, C
0.6

0.4

0.88

1

0.93

1

0.2
```

$p(t|C) = .3$
$p(r|C) = .7$
$p(o|V) = .1$
$p(e|V) = .3$
$p(y|V) = .4$
$p(r|V) = .2$

$\text{argmax}_{XYZ} p(ryy|XYZ) = ?$

Possible state seq.: (’,v)(v,c)(c,v)[VCV], (’,c)(c,c)(c,v)[CCV], (’,c)(c,v)(v,v) [CVV]
Viterbi Computation

\[ \begin{align*}
\alpha & = 1 \\
\alpha & = .6 \times .7 \\
& = .42 \\
\alpha & = .42 \times .12 \times .7 \\
& = .03528 \\
\alpha & = .42 \times .88 \times .2 \\
& = .07392 \\
\alpha & = .08 \times 1 \times .7 \\
& = .056 \\
\alpha & = .07392 \times .07 \times .4 \\
& = .002070 \\
\alpha_{max} & = .01792
\end{align*} \]

\[ \begin{align*}
p(t|C) & = .3 \\
p(r|C) & = .7 \\
p(o|V) & = .1 \\
p(e|V) & = .3 \\
p(y|V) & = .4 \\
p(r|V) & = .2
\end{align*} \]
**n-best State Sequences**

- Keep track of n best "back pointers":
- Ex.: n = 2:
  - Two "winners": VCV (best)
  - CCV (2\textsuperscript{nd} best)
Tracking Back the n-best paths

• Backtracking-style algorithm:
  • Start at the end, in the best of the n states (s\text{\textsubscript{best}})
  • Put the other n-1 best nodes/back pointer pairs on stack, except those leading from s\text{\textsubscript{\text{\text{best}}}} to the same best-back state.
• Follow the back “beam” towards the start of the data, spitting out nodes on the way (backwards of course) using always only the best back pointer.
• At every beam split, push the diverging node/back pointer pairs onto the stack (node/beam width is sufficient!).
• When you reach the start of data, close the path, and pop the top-most node/back pointer(width) pair from the stack.
• Repeat until the stack is empty; expand the result tree if necessary.
Pruning

- Sometimes, too many trellis states in a stage:

  \[
  \alpha = .002 \\
  \alpha = .043 \\
  \alpha = .001 \\
  \alpha = .231 \\
  \alpha = .0002 \\
  \alpha = .000003 \\
  \alpha = .000435 \\
  \alpha = .0066
  \]

criteria: 
(a) $\alpha < \text{threshold}$
(b) $\sum \pi < \text{threshold}$
(c) # of states $> \text{threshold}$
  (get rid of smallest $\alpha$)