A Primer on Hidden Markov Models

J.-C. Chappelier & M. Rajman

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectives/Contents

Objective:

- Introduce fundamental concepts necessary to use HMMs for PoS tagging

Contents:

- recap example
- HMM models, three basic problems
- Forward-Backward algorithms
- Viterbi algorithm
- Baum-Welch algorithm

J.-C. Chappelier & M. Rajman

Example: PoS tagging with HMM

Sentence to tag: Time flies like an arrow

Example of HMM model:

- PoS tags: $\mathcal{T} = \{\texttt{Adj}, \texttt{Adv}, \texttt{Det}, \texttt{N}, \texttt{V}, \ldots\}$
- Transition probabilities:

  $P(\texttt{N}|\texttt{Adj}) = 0.1, P(\texttt{V}|\texttt{N}) = 0.3, P(\texttt{Adv}|\texttt{N}) = 0.01, P(\texttt{Adv}|\texttt{V}) = 0.005,$
  $P(\texttt{Det}|\texttt{Adv}) = 0.1, P(\texttt{Det}|\texttt{V}) = 0.3, P(\texttt{N}|\texttt{Det}) = 0.5$

  (plus all the others, such that stochastic constraints are fulfilled)
- Initial probabilities:

  $P_I(\texttt{Adj}) = 0.01, P_I(\texttt{Adv}) = 0.001, P_I(\texttt{Det}) = 0.1, P_I(\texttt{N}) = 0.2, P_I(\texttt{V}) = 0.003$ (+...)
- ☆ Words: $\mathcal{L} = \{an, arrow, flies, like, time, \ldots\}$
- ☆ Emission probabilities:

  $P(time|\texttt{N}) = 0.1, P(time|\texttt{Adj}) = 0.01, P(time|\texttt{V}) = 0.05,$ (+...)
  $P(flies|\texttt{N}) = 0.1, P(flies|\texttt{V}) = 0.01, \quad P(like|\texttt{Adv}) = 0.005, P(like|\texttt{V}) = 0.1,$ (+...)
  $P(an|\texttt{Det}) = 0.3, \quad P(arrow|\texttt{N}) = 0.5$ (+...)

Example: PoS tagging with HMM (cont.)

In this example, $12 = 3 \cdot 2 \cdot 2 \cdot 1 \cdot 1$ analyzes are possible, for example:

$P(\textit{time}/\texttt{N}\ \textit{flies}/\texttt{V}\ \textit{like}/\texttt{Adv}\ \textit{an}/\texttt{Det}\ \textit{arrow}/\texttt{N}) = 1.13 \cdot 10^{-11}$
$P(\textit{time}/\texttt{Adj}\ \textit{flies}/\texttt{N}\ \textit{like}/\texttt{V}\ \textit{an}/\texttt{Det}\ \textit{arrow}/\texttt{N}) = 6.75 \cdot 10^{-10}$

Details of one of such computation:

$$
\begin{aligned}
& P(\textit{time}/\texttt{N}\ \textit{flies}/\texttt{V}\ \textit{like}/\texttt{Adv}\ \textit{an}/\texttt{Det}\ \textit{arrow}/\texttt{N}) \\
= \quad & P_I(\texttt{N}) \cdot P(\textit{time}|\texttt{N}) \cdot P(\texttt{V}|\texttt{N}) \cdot P(\textit{flies}|\texttt{V}) \cdot P(\textit{Adv}|\texttt{V}) \cdot P(\textit{like}|\texttt{Adv}) \\
& \cdot P(\texttt{Det}|\texttt{Adv}) \cdot P(\textit{an}/\texttt{Det}) \cdot P(\texttt{N}|\texttt{Det}) \cdot P(\textit{arrow}|\texttt{N}) \\
= \quad & \text{2e-1} \cdot \text{1e-1} \cdot \text{3e-1} \cdot \text{1e-1} \cdot \text{2e-2} \cdot \text{5e-3} \cdot \text{5e-3} \cdot \text{1e-1} \cdot \text{3e-1} \cdot \text{5e-1} \cdot \text{5e-1} \\
= \quad & 1.13 \cdot 10^{-11}
\end{aligned}
$$

The aim is to choose the most probable tagging among the possible ones (e.g. as provided by the lexicon)

J.-C. Chappelier & M. Rajman

Markov Models

Markov model: a discrete-time stochastic process **T** on $\mathcal{T} = \{t^{(1)}, ..., t^{(m)}\}$ satisfying the *Markov property* (limited conditioning):

$$P(T_i|T_1, ..., T_{i-1}) = P(T_i|T_{i-k}, ..., T_{i-1})$$

k : *order* of the Markov model

In practice $k = 1$ (bigrams) or 2 (trigrams) rarely 3 or 4 ($\rightarrow$ learning difficulties)

From a theoretical point of view: every Markov model of order k can be represented as another Markov model of order 1 (introduce $Y_i = (T_{i-k+1}, ..., T_i)$).

Vocable:

$$P(T_1, ..., T_i) = P(T_1) \cdot P(T_2|T_1) \cdot ... \cdot P(T_i|T_{i-1})$$

initial probabilities transition probabilities

J.-C. Chappelier & M. Rajman

Hidden Markov Models (HMM)

What is hidden?

☞ The model itself (i.e. the state sequence)

What do we see then?

☞ An *observation w* related to the state (but not the state itself)

Formally:

Example for PoS-tagging:

- a set of states $\mathcal{C} = \{C_1, ..., C_m\}$

  PoS tags $\mathcal{T} = \{t^{(1)}, ..., t^{(m)}\}$

- a transition probabilities matrix **A**:
  $A_{ij} = P(Y_{t+1} = C_j | Y_t = C_i)$, shorten $P(C_j|C_i)$

  $P(T_{i+1}|T_i)$

- an initial probabilities vector I:
  $I_i = P(Y_1 = C_i)$ or $P(Y_t = C_i|\text{“start”})$, shorten $P_I(C_i)$

  $P(T_1)$

- a set of “observables” Σ (not necessarily discrete, in general)

  words $\mathcal{L} = \{\omega^{(1)}, ..., \omega^{(L)}\}$

- m probability densities on Σ, one for each state (*emission probabilities*):
  $B_i(o) = P(X_t = o | Y_t = C_i)$ (for $o \in \Sigma$), shorten$P(o|C_i)$

  $P(w|T_i)$

Simple example of HMM

Example: a cheater tossing from two hidden (unfair) coins

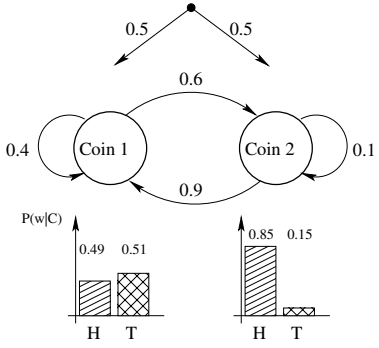
States: coin 1 and coin 2: $\mathcal{C} = \{1, 2\}$

transition matrix $\mathbf{A} = \begin{bmatrix} 0.4 & 0.6 \\ 0.9 & 0.1 \end{bmatrix}$

observed: Σ = {H, T}

emission probabilities:
$\mathbf{B_1} = (0.49, 0.51)$ and $\mathbf{B_2} = (0.85, 0.15)$

initial probabilities $\mathbf{I} = (0.5, 0.5)$



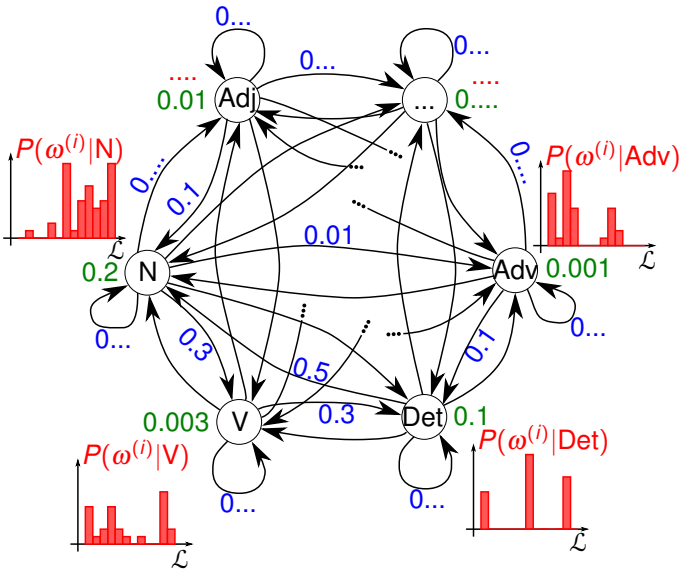
☞ 5 *free* parameters: I_1, A_{11}, A_{21}, $B_1(\mathtt{H})$, $B_2(\mathtt{H})$

Observation: HTTHTTHHTTHTTTHHTHHTTHTTTTHTHHTHTHHTTTH

[(Hidden) sequence of states: 211211211121112112111211112121121211112]

J.-C. Chappelier & M. Rajman

HMM example for PoS tagging



initial probabilities

transition probabilities

emission probabilities

The three basic problems for HMMs

Problems: Given an HMM and an observation sequence $\mathbf{w} = w_1 \ldots w_n$

- given the parameters $\boldsymbol{\theta}$ of the HMM, what is the probability of the observation sequence: $P(\mathbf{w}|\boldsymbol{\theta})$
  Application: Language Identification
- given the parameters $\boldsymbol{\theta}$ of the HMM, find the most likely state sequence $\mathbf{T} = T_1 \ldots T_n$ that produces $\mathbf{w}$: $\underset{\mathbf{T}}{\operatorname{argmax}}\, P(\mathbf{T}|\mathbf{w}, \boldsymbol{\theta})$
  Application: PoS Tagging, Speech recognition
- find the parameters that maximize the probability of producing $\mathbf{w}$: $\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(\boldsymbol{\theta}|\mathbf{w})$
  Application: Unsupervised learning

Remarks:

① $\boldsymbol{\theta} = (\mathbf{I}, \mathbf{A}, \mathbf{B})$
$= (I_1, ..., I_m, A_{11}, ..., A_{1m}, ..., A_{m1}, ..., A_{mm}, B_1(w_1), B_1(w_2), ..., B_1(w_L),$
$B_2(w_1), ..., B_2(w_L), ..., B_m(w_1), ..., B_m(w_L))$

i.e. $(m-1) + m \cdot (L-1) + m \cdot (m-1) = m \cdot (m+L-1) - 1$ *free* parameters (because of sum-to-1 contraints), where $m = |\mathcal{T}|$ and $L = |\mathcal{L}|$ (in the finite case, otherwise L stands for the total number of parameters used to represent $\mathcal{L}$)

② Supervised learning (i.e $\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(\boldsymbol{\theta}|\mathbf{w}, \mathbf{T})$) is easy

③ WARNING! There is a difference between $P(\boldsymbol{\theta}|\mathbf{w})$ and $P(\mathcal{M}|\mathbf{w})$!

The model $\mathcal{M}$ is supposed to be known here, but its parameters $\boldsymbol{\theta}$:
i.e. the HMM *design* is already done (number of states, alphabet) only the parameters are missing.

J.-C. Chappelier & M. Rajman

Contents

- ➥ HMM models, three basic problems
- ☞ Forward-Backward algorithms
- ➥ Viterbi algorithm
- ➥ Baum-Welch algorithm

Computation of $P(\mathbf{w}|\boldsymbol{\theta})$

Computation of $P(\mathbf{w}|\boldsymbol{\theta})$ is mathematically trivial:

$$P(\mathbf{w}|\boldsymbol{\theta}) = \sum_{\mathbf{T}} P(\mathbf{w}, \mathbf{T}|\boldsymbol{\theta}) = \sum_{\mathbf{T}} P(\mathbf{w}|\mathbf{T}, \boldsymbol{\theta}) \cdot P(\mathbf{T}|\boldsymbol{\theta})$$

Practical limitation: complexity is $\mathcal{O}(n m^n)$ $\rightsquigarrow$ exponential!

Practical computation: forward/backward algorithms $\longrightarrow$ complexity is $\mathcal{O}(n m^2)$

J.-C. Chappelier & M. Rajman

Forward-Backward algorithms

"forward" variable : $\alpha_i(t) = P(w_1, ..., w_i, T_i = t|\boldsymbol{\theta})$ $\quad t \in \mathcal{T}$

iterative computation: $\alpha_{i+1}(t') = B_{t'}(w_{i+1}) \cdot \sum_{t \in \mathcal{T}} (\alpha_i(t) \cdot A_{tt'})$

$\alpha_1(t) = B_t(w_1) \cdot I_t$

"backward" variable : $\beta_i(t) = P(w_{i+1}, ..., w_n | T_i = t, \boldsymbol{\theta})$

iterative computation: $\beta_{i-1}(t') = \sum_{t \in \mathcal{T}} (\beta_i(t) \cdot A_{t't} \cdot B_t(w_i))$

$\beta_n(t) = 1$ (by convention, practical considerations)

Computation in $\mathcal{O}(nm^2)$ $\rightarrow$ **efficient solutions to "first problem"**:

$$P(\mathbf{w}|\boldsymbol{\theta}) = \sum_{t \in \mathcal{T}} P(\mathbf{w}, T_n = t|\boldsymbol{\theta}) = \sum_{t \in \mathcal{T}} \alpha_n(t)$$

$$P(\mathbf{w}|\boldsymbol{\theta}) = \sum_{t \in \mathcal{T}} \alpha_i(t) \cdot \beta_i(t) \qquad \forall i : 1 \leq i \leq n$$

Forward-Backward algorithms (2)

There exist also

"forward-backward" variable : $\gamma_i(t) = P(T_i = t|\mathbf{w}, \boldsymbol{\theta})$

$$\gamma_i(t) = \frac{P(\mathbf{w}, T_i = t|\boldsymbol{\theta})}{P(\mathbf{w}|\boldsymbol{\theta})} = \frac{\alpha_i(t) \cdot \beta_i(t)}{\sum_{t' \in \mathcal{T}} \alpha_i(t') \cdot \beta_i(t')}$$

☞ useful later for the Baum-Welch algorithm

Contents

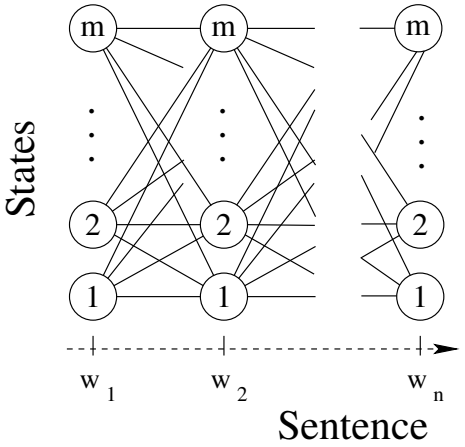
- ➥ HMM models, three basic problems
- ➥ Forward-Backward algorithms
- ☞ Viterbi algorithm
- ➥ Baum-Welch algorithm

Viterbi algorithm (1)

Efficient solution to the "second problem": find the most likely sequence of states **T** (knowing **w** and the parameters $\boldsymbol{\theta}$) : $\underset{\mathbf{T}}{\operatorname{argmax}} P(\mathbf{T}|\mathbf{w},\boldsymbol{\theta})$

$\Rightarrow$ maximize (in **T**) $P(\mathbf{T},\mathbf{w}|\boldsymbol{\theta})$.

"The" lattice ☞ temporal unfolding of all possible walks through the Markov chain

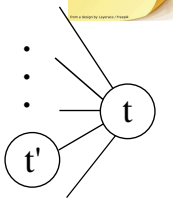


Viterbi algorithm (2)

Let $\rho_i(t) = \max_{T_1,...,T_{i-1}} P(T_1,...,T_{i-1},T_i=t,w_1,...,w_i|\boldsymbol{\theta})$

We are looking for $\max_{t\in\mathcal{T}} \rho_n(t)$

It can be shown (exercise) that $\rho_i(t) = \max_{t'}\left[P(t|t',\boldsymbol{\theta})\,P(w_i|t,\boldsymbol{\theta})\,\rho_{i-1}(t')\right]$



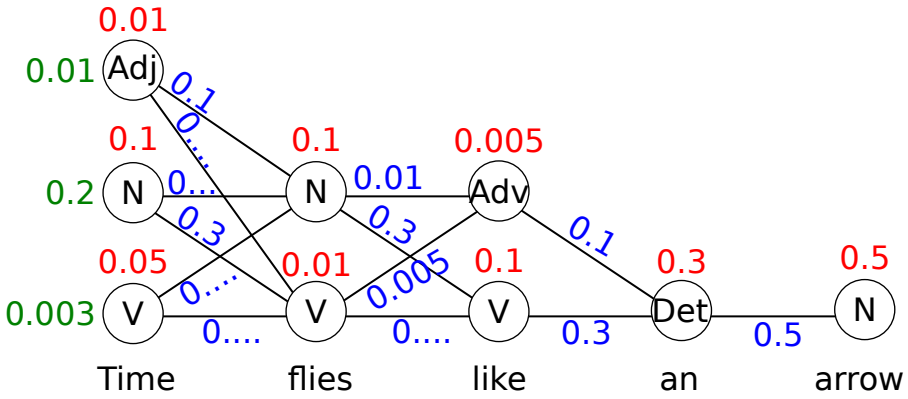
from which comes the following algorithm:

for all $t \in \mathcal{T}$ **do**
$\rho_1(t) = I_t \cdot B_t(w_1)$

for i from 2 to n **do**
for all $t \in \mathcal{T}$ **do**
- $\rho_i(t) = B_t(w_i) \cdot \max_{t'}\left(A_{t't} \cdot \rho_{i-1}(t')\right)$
- mark one of the transitions from t' to t where the maximum is reached

reconstruct backwards (from T_n) the best path following the marked transitions

Viterbi algorithm: example



Contents

- ➥ HMM models, three basic problems
- ➥ Forward-Backward algorithms
- ➥ Viterbi algorithm
- ☞ Baum-Welch algorithm

Expectation-Maximization

Our goal: maximize $P(\boldsymbol{\theta}|\mathbf{w})$

☞ Maximum-likelihood estimation of $\boldsymbol{\theta}$ → maximization of $P(\mathbf{w}|\boldsymbol{\theta})$

To achieve it: **Expectation-Maximization (EM) algorithm**

General formulation of EM: given

- observed data $\mathbf{w} = w_1 \ldots w_n$
- a parameterized probability distribution $P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})$ where
  - $\mathbf{T} = T_1 \ldots T_n$ are unobserved data
  - $\boldsymbol{\theta}$ are the parameters of the model

determine $\boldsymbol{\theta}$ that maximizes $P(\mathbf{w}|\boldsymbol{\theta})$ by convergence of iterative computation of the series $\boldsymbol{\theta}^{(i)}$ that maximizes (in $\boldsymbol{\theta}$) $\mathbf{E_T}\left[\log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})|\mathbf{w}, \boldsymbol{\theta}^{(i-1)}\right]$

Expectation-Maximization (2)

To do so, define the auxilary function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbf{E_T}\left[\log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})|\mathbf{w}, \boldsymbol{\theta}'\right] = \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{w}, \boldsymbol{\theta}') \log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})$$

as it can be shown (see Appendix) that

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') > Q(\boldsymbol{\theta}', \boldsymbol{\theta}') \Rightarrow P(\mathbf{w}|\boldsymbol{\theta}) > P(\mathbf{w}|\boldsymbol{\theta}')$$

This is the fundamental principle of EM:

if we already have an estimation $\boldsymbol{\theta}'$ of the parameters and we find another parameter configuration $\boldsymbol{\theta}$ for which the first inequality (on Q) holds,

then w is most probable with model $\boldsymbol{\theta}$ rather than with model $\boldsymbol{\theta}'$.

J.-C. Chappelier & M. Rajman

Expectation-Maximization (3)

EM algorithm:

- Expectation Step: Compute $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i)})$
- Maximization Step: Compute $\boldsymbol{\theta}^{(i+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i)})$

in other words:

1. Choose $\boldsymbol{\theta}^{(0)}$ (and set $i = 0$)
2. Find $\boldsymbol{\theta}^{(i+1)}$ which maximizes $\sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{w}, \boldsymbol{\theta}^{(i)}) \log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}^{(i+1)})$
3. Set $i \leftarrow i+1$ and go back to (2) unless some convergence test is fulfilled

Baum-Welch Algorithm

The Baum-Welch Algorithm is an EM algorithm for estimating HMM parameters.

It's an answer to the "third problem" (unsupervised learning).

The goal is therefore to find

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\mathbf{T}} P(\mathbf{T}|\mathbf{w}, \boldsymbol{\theta}') \log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\mathbf{T}} P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}') \log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})$$

since $P(\mathbf{w}|\boldsymbol{\theta}')$ does not depend on $\boldsymbol{\theta}$.

What is $\log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})$?

$$\log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}) = \log P_{\boldsymbol{l}}(T_1) + \sum_{i=2}^{n} \log P(T_i|T_{i-1}) + \sum_{i=1}^{n} \log P(w_i|T_i)$$

☞ It's maximization (see Appendix) leads to estimates $\widehat{l_t}$, $\widehat{A_{tt'}}$ and $\widehat{B_t(w)}$.

Baum-Welch Algorithm: effective computation

How do we compute these (re)estimates?

Let $\chi_i(t,t') = P(T_i = t, T_{i+1} = t' | \mathbf{w}, \boldsymbol{\theta})$

χ_i is easy to compute with "forward" and "backward" variables:

$$\chi_i(t,t') = \frac{\alpha_i(t) \cdot A_{tt'} \cdot B_{t'}(w_{i+1}) \cdot \beta_{i+1}(t')}{\sum_{\tau \in \mathcal{T}} \sum_{\tau' \in \mathcal{T}} \alpha_i(\tau) \cdot A_{\tau\tau'} \cdot B_{\tau'}(w_{i+1}) \cdot \beta_{i+1}(\tau')}$$

Notice: $\gamma_i(t) = \sum_{t' \in \mathcal{T}} \chi_i(t,t')$ for all $1 \leq i < n$

Effective reestimation formulas

$$\widehat{I_t} = \gamma_1(t)$$

$$\widehat{A_{tt'}} = \frac{\sum_{i=1}^{n-1} \chi_i(t,t')}{\sum_{i=1}^{n-1} \gamma_i(t)}$$

$$\widehat{B_t(w)} = \frac{\sum_{\substack{i=1 s.t. \\ w_i=w}}^{n} \gamma_i(t)}{\sum_{i=1}^{n} \gamma_i(t)} = \frac{\sum_{i=1}^{n} \gamma_i(t)\,\delta_{w_i,w}}{\sum_{i=1}^{n} \gamma_i(t)}$$

with $\delta_{w,w'} = 1$ if $w = w'$ and 0 otherwise.

Baum-Welch Algorithm

1. Let $\boldsymbol{\theta}^{(0)}$ be an initial parameter set
2. Compute iteratively α, β and then γ and χ
3. Compute $\boldsymbol{\theta}^{(t+1)}$ with reestimation formulas
4. If $\left|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\right| > \varepsilon$, go to (2) [or another weaker stop test]

WARNING!
The algorithm converges but only towards a **local** maximum of $\mathbf{E}\left[\log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta})\right]$

Other models

Beyond HMMs, what's next?

- Conditional Random Fields (CRF)
- Bayesian Networks
- Graphical Models

However, the three important main aspects remain:

1. efficient computations using dynamic programming
2. Viterbi-like search algorithm ("belief propagation")
3. Unsupervised learning with Expectation-Maximization

Keypoints

- HMMs definitions, their applications
- Three basic problems for HMMs
- Algorithms needed to solve these problems:
  - Forward-Backward
    (know what it solves and why it does exist, but not the mathematical details)
  - Viterbi
    (know everything and be able to do/apply it)
  - Baum-Welch
    (be aware of its existence and properties, but not the implementation details)

References

[1] L. R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, vol. 77, No. 2, 1989.

[2] C. D. Manning, H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT, 1999.

[3] A. P. Dempster, N. M. Laird, D. B. Rubin, *Maximum-likelihood from incomplete data via the EM algorithm*, Journal of Royal Statistical Society B, 1977.

[4] H. Bourlard et al., *Traitement de la parole*, 2000; pp. 179-200, 202-214, 232-260.

APPENDIX

EM: Justification of the maximization of the auxilary function Q

Finding $\boldsymbol{\theta}$ that maximises $P(\mathbf{w}|\boldsymbol{\theta})$ can be done by maximizing (in $\boldsymbol{\theta}$) $Q(\boldsymbol{\theta}, \boldsymbol{\theta}')$ (for any given $\boldsymbol{\theta}'$):

$$\log P(\mathbf{w}|\boldsymbol{\theta}) - \log P(\mathbf{w}|\boldsymbol{\theta}') = \log \frac{P(\mathbf{w}|\boldsymbol{\theta})}{P(\mathbf{w}|\boldsymbol{\theta}')} = \log \sum_{\mathbf{t}} \frac{P(\mathbf{w},\mathbf{t}|\boldsymbol{\theta})}{P(\mathbf{w}|\boldsymbol{\theta}')}$$

$$\begin{aligned}
&= \log \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{w},\boldsymbol{\theta}') \frac{P(\mathbf{w},\mathbf{t}|\boldsymbol{\theta})}{P(\mathbf{w},\mathbf{t}|\boldsymbol{\theta}')} \\
&\overset{\text{Jensen}}{\geq} \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{w},\boldsymbol{\theta}') \log \frac{P(\mathbf{w},\mathbf{t}|\boldsymbol{\theta})}{P(\mathbf{w},\mathbf{t}|\boldsymbol{\theta}')} \\
&\geq \mathbf{E}_{\mathbf{T}} \left[\log P(\mathbf{T},\mathbf{w}|\boldsymbol{\theta})|\mathbf{w},\boldsymbol{\theta}'\right] - \mathbf{E}_{\mathbf{T}} \left[\log P(\mathbf{T},\mathbf{w}|\boldsymbol{\theta}')|\mathbf{w},\boldsymbol{\theta}'\right] \\
&\geq Q(\boldsymbol{\theta},\boldsymbol{\theta}') - Q(\boldsymbol{\theta}',\boldsymbol{\theta}')
\end{aligned}$$

Therefore:

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}') > Q(\boldsymbol{\theta}',\boldsymbol{\theta}') \Rightarrow \log P(\mathbf{w}|\boldsymbol{\theta}) > \log P(\mathbf{w}|\boldsymbol{\theta}') \Rightarrow P(\mathbf{w}|\boldsymbol{\theta}) > P(\mathbf{w}|\boldsymbol{\theta}')$$

Baum-Welch algorithm: derivation of the formulas

Goal: Maximize

$$\widehat{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \log P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}) = \log P_I(T_1) + \sum_{i=2}^{n} \log P(T_i|T_{i-1}) + \sum_{i=1}^{n} \log P(w_i|T_i)$$

$\widehat{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ consists therefore of 3 terms:

$$\widehat{Q}((\mathbf{I}, \mathbf{A}, \mathbf{B}), \boldsymbol{\theta}') = Q_I(\mathbf{I}, \boldsymbol{\theta}') + Q_A(\mathbf{A}, \boldsymbol{\theta}') + Q_B(\mathbf{B}, \boldsymbol{\theta}')$$

Let's compute one of these:

$$\begin{aligned}
Q_I(\mathbf{I}, \boldsymbol{\theta}') &= \sum_{\mathbf{T}} P(\mathbf{T}, \mathbf{w}|\boldsymbol{\theta}') \log P_I(T_1) \\
&= \sum_{T_1} \sum_{T_2,\ldots,T_n} P(T_1, \mathbf{w}|\boldsymbol{\theta}') \cdot P(T_2, \ldots T_n|T_1, \mathbf{w}, \boldsymbol{\theta}') \cdot \log P_I(T_1) \\
&= \sum_{t \in \mathcal{T}} P(T_1 = t, \mathbf{w}|\boldsymbol{\theta}') \cdot \log P_I(t) \underbrace{\sum_{T_2,\ldots,T_n} P(T_2, \ldots, T_n|T_1, \mathbf{w}, \boldsymbol{\theta}')}_{=1} \\
&= \sum_{t \in \mathcal{T}} P(T_1 = t, \mathbf{w}|\boldsymbol{\theta}') \cdot \log l_t
\end{aligned}$$

J.-C. Chappelier & M. Rajman

Similarly we have:

$$Q_A(\mathbf{A}, \boldsymbol{\theta}') = \sum_{i=2}^{n} \sum_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}} P(T_{i-1} = t, T_i = t', \mathbf{w} | \boldsymbol{\theta}') \log A_{tt'}$$

$$Q_B(\mathbf{B}, \boldsymbol{\theta}') = \sum_{i=1}^{n} \sum_{t \in \mathcal{T}} P(T_i = t, \mathbf{w} | \boldsymbol{\theta}') \log B_t(w_i)$$

Therefore $\widehat{Q}$ is a sum of three **independent** terms (e.g. Q_I does not depend on $\mathbf{A}$ nor on $\mathbf{B}$)

therefore the maximisation over $\boldsymbol{\theta}$ is achieved by the three terms separately, i.e. maximizing $Q_I(\mathbf{I}, \boldsymbol{\theta}')$ over $\mathbf{I}$, $Q_A(\mathbf{A}, \boldsymbol{\theta}')$ over $\mathbf{A}$ and $Q_B(\mathbf{B}, \boldsymbol{\theta}')$ over $\mathbf{B}$ separately.

Notice that all these three functions are sums (over i) of functions of the form:

$$f(\mathbf{x}) = \sum_{j=1}^{m} y_j \log x_j$$

and all the above three functions have to be maximized under the constraint $\sum_{j=1}^{m} x_j = 1$.[1]

[1] To be accurate: for $\mathbf{B_t}$ the constraint is $\sum_{w \in \mathcal{L}} B_t(w) = 1$. This changes the formulas a bit, but not the essence of the computation.

J.-C. Chappelier & M. Rajman

Maximizing

$$f(\mathbf{x}) = \sum_{j=1}^{m} y_j \log x_j$$

under the constraint

$$\sum_{j=1}^{m} x_j = 1$$

can be achieved using Lagrange multipliers, i.e. looking at

$$g(\mathbf{x}) = f(\mathbf{x}) - \lambda \cdot \sum_{j=1}^{m} x_j = \sum_{j=1}^{m} \left(y_j \log x_j - \lambda \cdot x_j \right)$$

Solving this by $\frac{\partial}{\partial x} g(x) = 0$, we find that $\lambda = \frac{y_j}{x_j}$.
Putting this back in the constraint we find:

$$x_j = \frac{y_j}{\sum_{j=1}^{m} y_j}$$

Summarizing the obtained results, we have the following reestimation formulas (where the max. is reached):

$$\widehat{I_t} = \frac{P(T_1 = t, \mathbf{w}|\boldsymbol{\theta}')}{\sum_{t' \in \mathcal{T}} P(T_1 = t', \mathbf{w}|\boldsymbol{\theta}')} = \frac{P(T_1 = t, \mathbf{w}|\boldsymbol{\theta}')}{P(\mathbf{w}|\boldsymbol{\theta}')}$$

$$\widehat{A_{tt'}} = \frac{\sum_{i=2}^{n} P(T_{i-1} = t, T_i = t', \mathbf{w}|\boldsymbol{\theta}')}{\sum_{i=2}^{n} \sum_{\tau \in \mathcal{T}} P(T_{i-1} = t, T_i = \tau, \mathbf{w}|\boldsymbol{\theta}')}$$

$$= \frac{\sum_{i=2}^{n} P(T_{i-1} = t, T_i = t', \mathbf{w}|\boldsymbol{\theta}')}{\sum_{i=2}^{n} P(T_{i-1} = t, \mathbf{w}|\boldsymbol{\theta}')}$$

and:

$$\widehat{B_t(w)} = \frac{\sum_{\substack{i=1 s.t. \\ w_i=w}}^{n} P(T_i = t, \mathbf{w}|\boldsymbol{\theta}')}{\sum_{i=1}^{n} P(T_i = t, \mathbf{w}|\boldsymbol{\theta}')} = \frac{\sum_{i=1}^{n} P(T_i = t, \mathbf{w}|\boldsymbol{\theta}')\,\delta_{w_i,w}}{\sum_{i=1}^{n} P(T_i = t, \mathbf{w}|\boldsymbol{\theta}')}$$

with $\delta_{w,w'} = 1$ if $w = w'$ and 0 otherwise.

J.-C. Chappelier & M. Rajman