Hidden Markov Models

- Probabilistic graphical model
- Generative Approach
- Fully connected: "ergodic" HMM ; "Left-Right" HMM: no backlinks

Methods to work with HMMs

1.) Matching (FORWARD - / BACKWARD Algorithm)
$\alpha_t(s_i)$ $\beta_t(s_i)$

find: $p(\langle o_1, \dots, o_n \mid \lambda)$

2.) Most likely sequence (of states) for given observation

$$\underset{\langle s_1, \dots, s_n \rangle}{\operatorname{argmax}} \; p(\langle o_1, \dots, o_n \rangle, \langle s_1, \dots, s_n \rangle)$$

Viterbi Algorithm

3.) Training: find $\lambda = (A, B, \vec{\pi})$ given training samples

Baum-Welch-Formulas

- FORWARD / BACKWARD
- EM-Algorithm

PA Summary (2) HMMs 26.07.19

FORWARD Algorithm

Forward Variable: $\alpha_t(i) = P(o_1, \dots, o_t, q_t = s_i \mid \lambda)$

$o_1, \dots, o_t$: partial sequence observation — $q_t = s_i$: state s_i at time t

Initialization: $\alpha_1(i) = \pi_i \, b_i(o_1)$

Induction: $\alpha_{t+1}(i) = \left(\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \right) \cdot b_j(o_{t+1})$

$b_j(o_{t+1})$: observation probability of o_{t+1} at state j

Termination: $P(o \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Computation requires $N^2 T$ calculations

BACKWARD Algorithm

Backward variable: $\beta_t(i) = P(o_{t+1}, \dots, o_T \mid q_t = s_i, \lambda)$

$o_{t+1}, \dots, o_T$: partial sequence observation — $q_t = s_i$: given state i at time t

Initialization: $\beta_T(i) = 1$ (choosen arbitrarily)

Induction: $\beta_t(i) = \sum_{i=1}^{N} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)$

Computation requires $N^2 T$ calculations

Solution for Task 2 (Most likely hidden state sequence)

Individually most likely states:

$$\gamma_t(i) = P(q_t = s_i \mid O, \lambda) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\,\beta_t(i)}$$

$$q_t = \underset{1 \le i \le N}{\operatorname{argmax}} \; \gamma_t(i)$$

(state with highest probability)

Problem: resulting state sequence might not even be possible ...

⇒ Define best state sequence $P(Q \mid O, \lambda)$, which is equivalent to $P(Q, O \mid \lambda)$ maximization

Viterbi Algorithm

Same as FORWARD but:

- Replace every Σ (sum) by max
- Technical detail: store actual path in separate list ψ.

Solution for task 3: Training / Learning

Determine $A, B, \vec{\pi}$: no analytical solution $\Rightarrow$ solve iteratively

Baum-Welch-Formulas (FORWARD / BACKWARD + EM)

Expectation

$\xi_t(s_i, s_j)$: Probability for transition $s_i \to s_j$ at time t

$$\xi_t(s_i, s_j) = \frac{\alpha_t(s_i) \cdot a_{s_i s_j} \cdot b_{s_j}(o_{t+1}) \cdot \beta_{t+1}(s_j)}{\sum_{k=1}^{M} \sum_{l=1}^{M} \alpha_t(s_k) \cdot a_{s_k s_l} \cdot b_{s_l}(o_{t+1}) \cdot \beta_{t+1}(s_l)}$$

normalization for all transitions between time t and $t+1$ $\Rightarrow$ ξ is a PDF

$\gamma_t(s_i)$: Probability of being in state s_i at time t

$$\gamma_t(s_i) = \sum_{s_j=1}^{M} \xi_t(s_i, s_j)$$

$\sum_{t=1}^{T} \xi_t(s_i, s_j)$: $E\#(s_i, s_j)$ $(s_i \to s_j)$

$\sum_{t=1}^{T} \gamma_t(s_i)$: $E\#(s_i)$ (from s_i)

PA Summary ⑥ HMMs / MRF 26.07.19

Maximization

$\overline{\pi_i}$: expected # of times in s_i at $t=1$

$$\overline{\pi_i} = \gamma_1(s_i)$$

$$\overline{a_{ij}} = \frac{E\#(s_i, s_j)}{E\#(s_i)}$$

$$\overline{b_{s_j}(k)} = \frac{E\#(s_j) \quad s.t. \quad o_t = k}{E\#(s_j)}$$

Markov Random Field

$$[f_{ij}] = \underset{f_{ij}}{\operatorname{argmax}} \; P([g_{ij}] \mid [f_{ij}]) \cdot P([f_{ij}])$$

$$P(f_{ij} \mid f_{11}, \dots, f_{NM}) = P(f_{ij} \mid N(f_{ij}))$$

Hammersley-Clifford Theorem : MRF $\Leftrightarrow$ GRF

$$p(\vec{x}) = \frac{1}{Z} \cdot e^{-H(\vec{x})} \qquad H(\vec{x}) = \sum_{m \in S} V_m(\vec{x})$$

with submodularity condition: Define V_m s.t. similar vectors yield low Energy values

$$E(0,0) + E(1,1) \leq E(0,1) + E(1,0)$$

For our image smoothing example:

Pairwise Potential ("regularizer")

$$E(f_{ij}, f' \in N(f_{ij})) = \|f_{ij} - f'\|_2^2$$

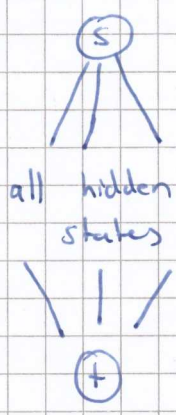
Unary Potential ("data term")

$$-\log\left[N(g_{ij}, f_{ij}, \sigma)\right) = \frac{(f_{ij} - g_{ij})^2}{\sigma}$$

Minimization Problem

$$\sum_{i=1}^{N} \left(E(x_i) + \sum_{j=1}^{\tilde{N}} E(x_i, x_j) \right)$$

Graph Cuts for Energy minimization



Edge (S) — (x_{ij}) assignment ∅ to x_{ij}

(x_{ij}) — (T) assignment 1 to x_{ij} [if included in cut]

Cut cost (for each edge cut):

Energy value unary / pairwise

⟹ min cut finds x assignment for unary and pairwise potentials that leads to overall lowest Energy

⇒ highest $P(x_{ij})$ for the MRF

⇒ solution (for binary labels)

α - Expansion Algorithm for L labels:

while (changes):

for each label l:

binaryGraphCut(l) ⇐ 1 vs. all Graph cut:

"does the number of assignments for label l grow?

Iterate until everything is stable