

CS195-5 : Introduction to Machine Learning

Lecture 33

Greg Shakhnarovich

December 4, 2006

Announcements

Review: inference with HMM

- Inference regarding observation sequence $\mathbf{x}_1, \dots, \mathbf{x}_N$
 - Compute likelihood of a model given observations
⇒ the forward-backward algorithm
 - Sample from the model
- Inference regarding hidden states
 - Estimate most likely state sequence
⇒ the Viterbi algorithm
- Estimating the model parameters
⇒ EM (Baum-Welch algorithm)

Review: the forward-backward algorithm

- Forward and backward probabilities:

$$\alpha_t(s) \triangleq p(\mathbf{x}_1, \dots, \mathbf{x}_t, s_t = s), \beta_t(s) \triangleq p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_N | s_t = s)$$

- The forward-backward algorithm:

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s),$$

$$\alpha_t(s) = \left[\sum_{s'} \alpha_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s)$$

$$\beta_N(s) = 1$$

$$\beta_t(s) = \sum_{s'} [p(s \rightarrow s')p(\mathbf{x}_{t+1} | s_{t+1} = s') \beta_{t+1}(s')]$$

Use of forward-backward probabilities

- Likelihood of observations:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_s \alpha_t(s) \beta_t(s)$$

- State posterior:

$$\gamma_t(s) \triangleq p(s_t = s \mid \mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{\alpha_t(s) \beta_t(s)}{\sum_{s'} \alpha_t(s') \beta_t(s')}$$

- Transition posterior:

$$\begin{aligned} \xi_t(i, j) &\triangleq p(s_t = i, s_{t+1} = j \mid \mathbf{x}_1, \dots, \mathbf{x}_N) \\ &= \frac{\alpha_t(i) p(s_{t+1} = j \mid s_t = i) p(\mathbf{x}_{t+1} \mid s_{t+1} = j) \beta_{t+1}(j)}{\sum_{s'} \alpha_t(s') \beta_t(s')} \end{aligned}$$

Review: the Baum-Welch (EM) algorithm

- **E-step:** estimate $\gamma_t^{(l)}(s)$, $\xi_t^{(l)}(i, j)$ using θ^{old}
- **M-step:** estimate θ^{new} using γ and ξ .

$$p^{new}(s_1 = s) = \frac{1}{L} \sum_{l=1}^L \gamma_1^{(l)}(s),$$

$$\hat{n}_{i,j} = \sum_{l=1}^L \sum_{t=1}^{n_l-1} \xi_t^{(l)}(i, j), \quad \Rightarrow \quad p^{new}(i \rightarrow j) = \frac{\hat{n}_{i,j}}{\sum_{j'} \hat{n}_{i,j'}}$$

$$\theta_s^{new} = \operatorname{argmin}_{\theta_s} \sum_{l=1}^L \sum_{t=1}^{n_l-1} \gamma_t^{(l)}(s) \log p(\mathbf{x}_t^{(l)}; \theta_s)$$

HMM: scaling

- In naive implementation, α_t and β_t will underflow.
- Solution: scaling by $c_t \triangleq p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s), \Rightarrow \tilde{\alpha}_1(s) = \alpha_1(s) / c_t,$$

$$\tilde{\alpha}_t(s) = \left[\sum_{s'} \tilde{\alpha}_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s) / c_t$$

- Note:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1})$$

HMM: scaling

- In naive implementation, α_t and β_t will underflow.
- Solution: scaling by $c_t \triangleq p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s), \Rightarrow \tilde{\alpha}_1(s) = \alpha_1(s) / c_t,$$

$$\tilde{\alpha}_t(s) = \left[\sum_{s'} \tilde{\alpha}_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s) / c_t$$

- Note:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = \prod_{i=1}^t c_i.$$

HMM: scaling

- In naive implementation, α_t and β_t will underflow.
- Solution: scaling by $c_t \triangleq p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s), \Rightarrow \tilde{\alpha}_1(s) = \alpha_1(s) / c_t,$$

$$\tilde{\alpha}_t(s) = \left[\sum_{s'} \tilde{\alpha}_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s) / c_t$$

- Note:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = \prod_{i=1}^t c_i.$$

- Also, $\tilde{\alpha}_1(s) = \alpha_1(s)/c_1$;

HMM: scaling

- In naive implementation, α_t and β_t will underflow.
- Solution: scaling by $c_t \triangleq p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s), \Rightarrow \tilde{\alpha}_1(s) = \alpha_1(s) / c_t,$$

$$\tilde{\alpha}_t(s) = \left[\sum_{s'} \tilde{\alpha}_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s) / c_t$$

- Note:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = \prod_{i=1}^t c_i.$$

- Also, $\tilde{\alpha}_1(s) = \alpha_1(s)/c_1$; $\tilde{\alpha}_2(s) = \alpha_2(s)/c_1c_2$;

HMM: scaling

- In naive implementation, α_t and β_t will underflow.
- Solution: scaling by $c_t \triangleq p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$

$$\alpha_1(s) = p_0(s)p(\mathbf{x}_1 | s_1 = s), \Rightarrow \tilde{\alpha}_1(s) = \alpha_1(s) / c_t,$$

$$\tilde{\alpha}_t(s) = \left[\sum_{s'} \tilde{\alpha}_{t-1}(s')p(s' \rightarrow s) \right] p(\mathbf{x}_t | s_t = s) / c_t$$

- Note:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = \prod_{i=1}^t c_i.$$

- Also, $\tilde{\alpha}_1(s) = \alpha_1(s)/c_1$; $\tilde{\alpha}_2(s) = \alpha_2(s)/c_1c_2$; \dots , $\tilde{\alpha}_t(s) = \alpha_t(s)/\prod_{i=1}^t c_i$.

Scaled forward probabilities

$$c_t = p(\mathbf{x}_t | \mathbf{x}_1, \dots, \mathbf{x}_{t-1}), \quad p(\mathbf{x}_1, \dots, \mathbf{x}_t) = \prod_{i=1}^t c_i$$

- The scaled α s are

$$\begin{aligned} \tilde{\alpha}_t(s) &= \frac{\alpha_t(s)}{c_1 \cdots c_t} = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_t, s_t = s)}{p(\mathbf{x}_1, \dots, \mathbf{x}_t)} \\ &= p(s_t = s | \mathbf{x}_1, \dots, \mathbf{x}_t). \end{aligned}$$

- Need to compute and store c_t in forward pass
 - Easy: c_t are the values such that $\sum_s \tilde{\alpha}_t(s) = 1$.

Scaled backward probabilities

- We also normalize β s:

$$\tilde{\beta}_t(s) = \frac{\beta_t(s)}{\prod_{i=t+1}^N c_i}.$$

- Recursion (the backward pass) becomes

$$\tilde{\beta}_t(s) = \sum_{s'} \left[p(s \rightarrow s') p(\mathbf{x}_{t+1} \mid s_{t+1} = s') \tilde{\beta}_{t+1}(s') \right] / c_{t+1}$$

- The scaled probability interpretation:

$$\tilde{\beta}_t(s) = \frac{p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_N \mid s_t = s)}{p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_N \mid \mathbf{x}_1, \dots, \mathbf{x}_t)}$$

Scaled likelihoods and posteriors

- Likelihood from c_t :

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{t=1}^N c_t.$$

- State posteriors:

$$\gamma_t(s) = \tilde{\alpha}_t(s)\tilde{\beta}_t(s).$$

- Transition posteriors:

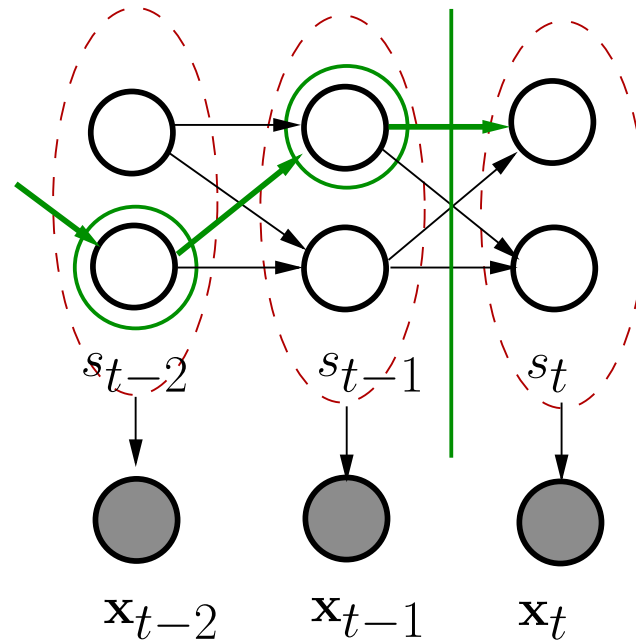
$$\xi_t(i, j) = c_{t+1}\tilde{\alpha}_t(s)p(s_{t+1} = j | s_t = i)p(\mathbf{x}_{t+1} | s_{t+1} = j)\tilde{\beta}_{t+1}(j).$$

We can verify these equations from expressions for $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$, γ_t , ξ_t and $\tilde{\alpha}_t$, $\tilde{\beta}_t$.

Inference with HMM

- Inference regarding observation sequence $\mathbf{x}_1, \dots, \mathbf{x}_N$
 - Compute likelihood of a model given observations
⇒ the forward-backward algorithm
 - Sample from the model
- Inference regarding hidden states
 - Estimate most likely state sequence
⇒ the Viterbi algorithm
- Estimating the model parameters
⇒ EM (Baum-Welch algorithm)

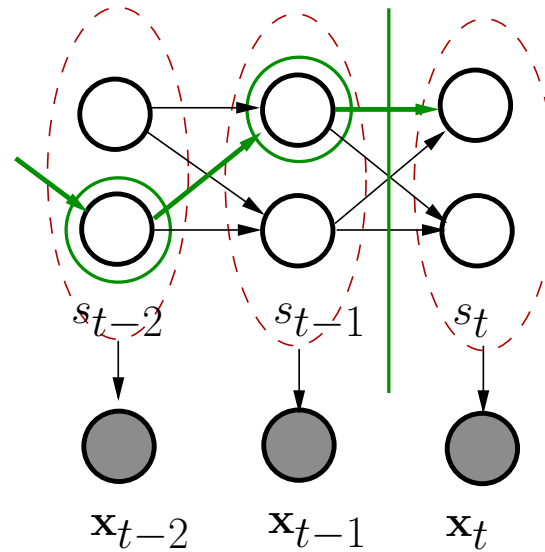
Most likely states



- We can define *max-probabilities*:

$$\delta_t(s) \triangleq \max_{s_1, \dots, s_{t-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_t, s_1, \dots, s_{t-1}, s_t = s)$$

Max-probabilities recursion



$$\delta_1(s) = p(s_1 = s)p(\mathbf{x}_1 | s_1 = s),$$

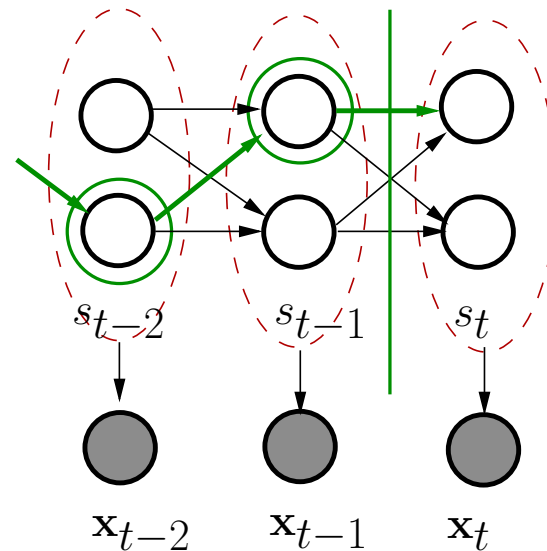
$$\delta_t(s) = \max_{s'} [\delta_{t-1}(s')p(s' \rightarrow s)] p(\mathbf{x}_t | s_t = s)$$

Viterbi decoding algorithm

$$\delta_t(s) = \max_{s_1, \dots, s_{t-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_t, s_1, \dots, s_{t-1}, s_t = s)$$

- We can recover the *most likely sequence of states* working backwards:

$$s_N^* = \operatorname{argmax}_s \delta_N(s),$$



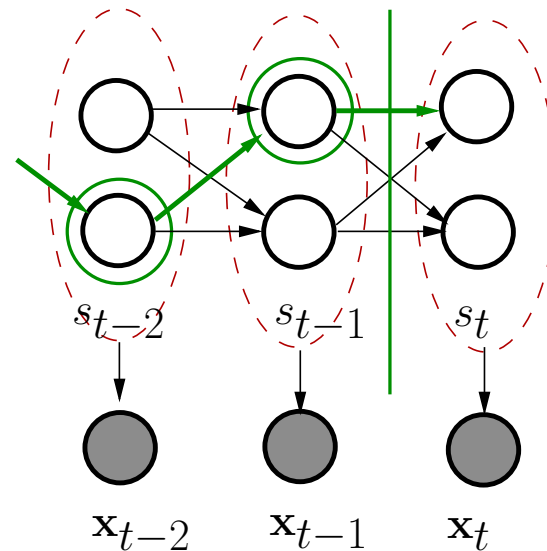
Viterbi decoding algorithm

$$\delta_t(s) = \max_{s_1, \dots, s_{t-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_t, s_1, \dots, s_{t-1}, s_t = s)$$

- We can recover the *most likely sequence of states* working backwards:

$$s_N^* = \operatorname{argmax}_s \delta_N(s),$$

$$s_t^* = \operatorname{argmax}_s \delta_t(s) p(s \rightarrow s_{t+1}^*).$$



- Note: not the same as $\operatorname{argmax}_s \gamma_t(s)$!

Viterbi decoding: properties

- Similar to forward-backward (sum-product) but instead of the sum uses max.
 - Bugs: at each time step, kill all the bugs except for the one with the largest value.
- Suppose s_1^*, \dots, a_N^* is the Viterbi path.
- Then, if $s_t^* = s$, then s_1^*, \dots, s_t^* is the most likely path *ending* at s_t^* , given only $\mathbf{x}_1, \dots, \mathbf{x}_t$.
- Decoding applications: \mathbf{x}_t is the noise dependent on s_t .
 - Used in phone-line modems, digital TV broadcasting, and in more than 10^9 cell phones today.

HMM as a generative model

- We can sample a sequence from HMM.
 - Draw $s_1 \sim p(s_1)$
 - Draw $\mathbf{x}_1 \sim p(\mathbf{x}; \theta_{s_1})$.
 - For $t = 2, \dots, N$, draw s_t, \mathbf{x}_t the same way.
- When we are done: $\mathbf{x}_1, \dots, \mathbf{x}_N$ is sampled from the model
 - And so is s_1, \dots, s_N .

Classification with HMM

- Training data: sequences labeled with class identity.
- Learn HMM parameters θ_y per class y using Baum-Welch.
- Standard generative model classifier:
- For a test sequence $X = \mathbf{x}_1, \dots, \mathbf{x}_N$, compute for each class y

$$\ell(X; y) = \log p(X; \theta_y)$$

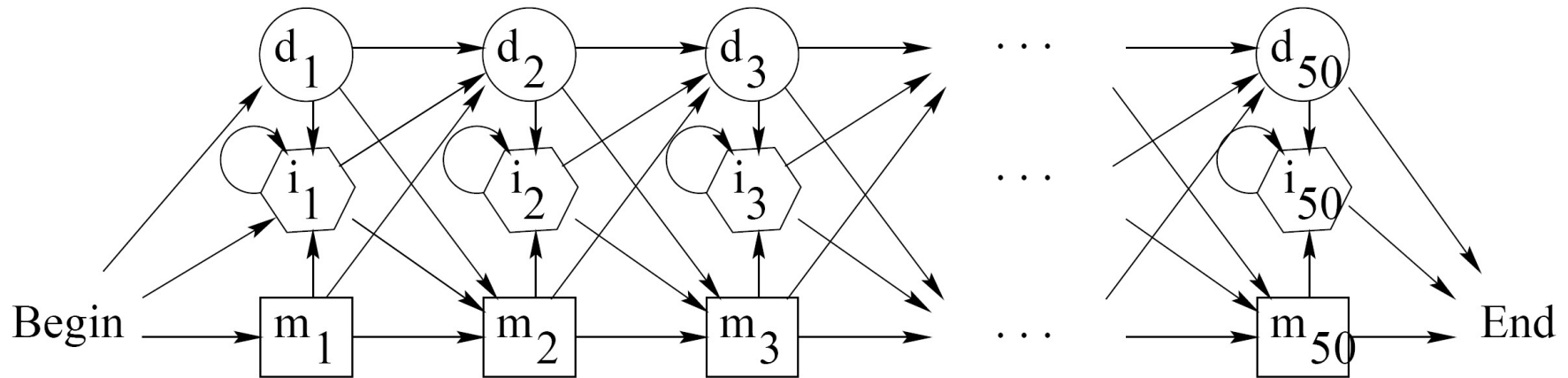
Classification with HMM

- Training data: sequences labeled with class identity.
- Learn HMM parameters θ_y per class y using Baum-Welch.
- Standard generative model classifier:
- For a test sequence $X = \mathbf{x}_1, \dots, \mathbf{x}_N$, compute for each class y

$$\ell(X; y) = \log p(X; \theta_y)$$

$$y^* = \operatorname{argmax}_y \log p(y) + \log p(X; \theta_y)$$

String-edit HMMs



- Modeling sequences of symbols, corresponding to template and potentially corrupted by noise.
- Three kinds of states:

m_t match the observed symbol x_t ,

d_t delete the symbol at position t ,

i_t insert a symbol at position t .

String-edit HMMs

- Also called “profile HMM”.
- Used in modeling protein sequences.
 - Template:

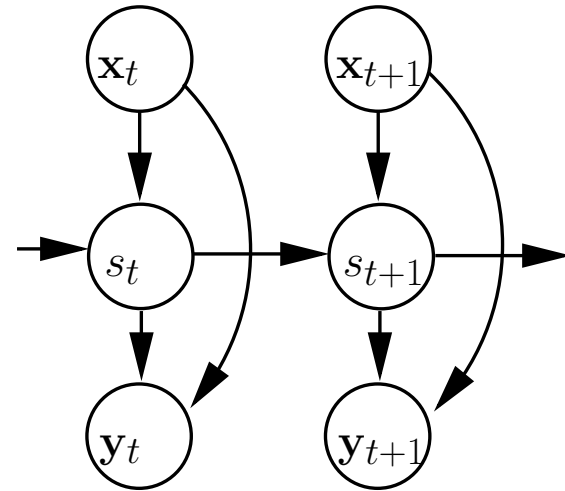
GVAALGAKVLAQIGVAVSHLGDEGKMVAQMKA VGVRHKGYGNKHIKAQYFEPLGASLLSAME...

- Observed:

GVAALGAKVLAQIGVAVSHLGD~~egk~~MVAQMKA VGVRHK~~gyg~~NK-HIKAQYFEPLGASLLSAME..

Input-Output HMM

- Model a state-dependent mapping from *input* \mathbf{x}_t to output \mathbf{y}_t .



$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{y}_1, \dots, \mathbf{y}_N, s_1, \dots, s_N) = p(\mathbf{x}_1) p(s_1 | \mathbf{x}_1) p(\mathbf{y}_1 | \mathbf{x}_1, s_1) \\ \times \prod_{t=2}^N p(\mathbf{x}_t) p(s_t | \mathbf{x}_t, s_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t, s_t)$$

- Can be seen as a “recurrent mixture of experts” model.