Short course

A vademecum of statistical pattern recognition and machine learning

The hidden Markov model

Massimo Piccardi
University of Technology, Sydney, Australia

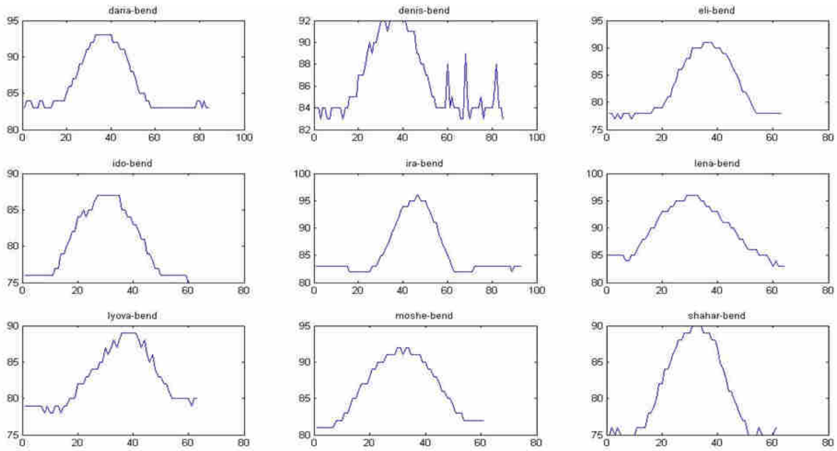
Agenda

- Sequential data
- Markov chain
- Hidden Markov model
- Evaluation, inference, decoding and estimation
- The forward, backward and forward/backward algortihms
- The Viterbi algorithm
- The Baum-Welch algorithm
- Continuous observations
- Sampling
- References

Sequential data

- So far, data from each class were assumed drawn from a single generating distribution, independently of each other (i.i.d. assumption)
- **Sequential data**, instead, are data sequences, not sets; the order is important
- Each sample may be generated from a different distribution
- Examples: over time: rainfall data; over space: nucleotide pairs in DNA

Example



- Approximate position of the head of a person during a bending action: x-axis: time frame; y-axis: distance in pixels from the top of the frame; 9 subjects

Markov chain

- A simple model to describe the evolution of a system
- Let us assume that the systems evolves **over time**, in discrete steps labelled 1 (initial time)...t...T (final time)
- Let us have T discrete variables, $q_1 \ldots q_t \ldots q_T$, each taking value in a finite set of N symbols, $S = \{s_1, s_2, \ldots, s_N\}$
- Let us also assume the (first-order) *Markov hypothesis*:

$$p(q_t = s_j \mid q_{t-1} = s_i, q_{t-2} = s_k, \ldots) = p(q_t = s_j \mid q_{t-1} = s_i)$$

 5

Markov chain: graphical model

- A *graphical model* represents graphically the conditional independencies/dependencies in a set of random variables
- Graphical model for the Markov chain:

$$q_1 \rightarrow q_2 \;\cdots\; \rightarrow q_{t-1} \rightarrow q_t \;\cdots\; \rightarrow q_{T-1} \rightarrow q_T$$

- Variables $q_1 \ldots q_t \ldots q_T$ are often referred to as *states* of the Markov chain
- The Markov chain can also be referred to as *discrete Markov process*

 6

Joint probability in a Markov chain

- Joint probability of the states over T frames:

$$p(Q) = p(q_1,\ldots,q_t,\ldots,q_T)$$

General case (chain rule)	$p(q_1,\ldots,q_t,\ldots,q_T) =$ $p(q_T \mid q_1,\ldots,q_{T-1})\,p(q_1,\ldots,q_{T-1}) =$ $p(q_T \mid q_1,\ldots,q_{T-1})\,p(q_{T-1} \mid q_1,\ldots,q_{T-2})\,p(q_1,\ldots,q_{T-2}) = \ldots$
Markov chain!	$p(q_1,\ldots,q_t,\ldots,q_T) =$ $p(q_T \mid q_{T-1})\,p(q_1,\ldots,q_{T-1}) =$ $p(q_T \mid q_{T-1})\,p(q_{T-1} \mid q_{T-2})\,p(q_1,\ldots,q_{T-2}) = \ldots =$ $p(q_1)\prod_{t=2}^{T} p(q_t \mid q_{t-1})$

Markov chain parameters

- Let us assume that the conditional probabilities are independent of the time (stationary or time-invariant model)
- This means that each probability:

$$p(q_t = s_j \mid q_{t-1} = s_i)$$

depends only on indices i and j (a contingency table with NxN combinations) and not on t

- We can introduce short-hand notations such as:

$$a_{ij} = a_{q_{t-1}=i\; q_t=j} = p(q_t = s_j \mid q_{t-1} = s_i)$$

Markov chain parameters

- The conditional probabilities are also known as the *state transition probabilities* and are the main parameters of the Markov chain; together, they form the N x N state transition matrix, A
- To complete the parameters, we also need the *initial state probabilities*:

$$\pi_i = \pi_{q_1=i} = p\left(q_1 = s_j\right)$$

- The parameters fully specify the model, allowing computation of the joint probability, all marginals and other conditionals

Markov chain: other properties

- One can re-write the chain backwards:

$$\begin{aligned}
&p(q_1,\dots,q_t,\dots,q_T) = \\
&p(q_T \mid q_{T-1})\dots p(q_3 \mid q_2)p(q_2 \mid q_1)p(q_1) = \\
&p(q_T \mid q_{T-1})\dots p(q_3 \mid q_2)p(q_2, q_1) = \\
&p(q_T \mid q_{T-1})\dots p(q_3 \mid q_2)p(q_2)p(q_1 \mid q_2) = \\
&p(q_T \mid q_{T-1})\dots p(q_3)p(q_2 \mid q_3)p(q_1 \mid q_2) = \\
&p(q_T)\prod_{t=1}^{T-1} p(q_t \mid q_{t+1})
\end{aligned}$$

- Comparing with the general chain rule in reverse order:

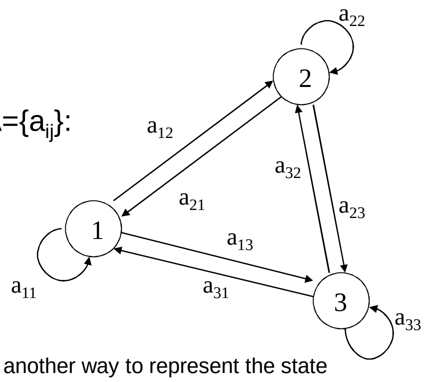
$$p(q_t \mid q_{t+1},\dots,q_T) = p(q_t \mid q_{t+1})$$

A simple example

- A simple three-state Markov chain of the weather
  - s_1 = "rainy"
  - s_2 = "cloudy"
  - s_3 = "sunny"

- State transition matrix $A=\{a_{ij}\}$:

0.4	0.3	0.3
0.2	0.6	0.2
0.1	0.1	0.8



The *state transition diagram* is another way to represent the state transition matrix (not to be confused with the graphical model)

Examples

Question 1

- Given that the weather on day t = 1 is sunny, what is the probability that the weather for the next 7 days will be "sun, sun, rain, rain, sun, clouds, sun"?

Answer:

$p(q_2=s_3,q_3=s_3,q_4=s_1,q_5=s_1,q_6=s_3,q_7=s_2,q_8=s_3|q_1=s_3) =$
$= a_{33}\ a_{33}\ a_{31}\ a_{11}\ a_{13}\ a_{32}\ a_{23} = 0.8 * 0.8 * 0.1 * 0.4 * 0.3 * 0.1 * 0.2$

Question 2

- What is the probability that the weather stays in the same known state s_i for exactly T consecutive days?

Answer:

$a_{ii}^{T-1}(1 - a_{ii})$

examples courtesy of Gutierrez-Osuna; and Rabiner, Juang

Examples

Question 3

- What is the probability that the state in t = 3 is, say, cloudy?

Answer:

The above is a marginal probability, $p(q_3 = s_2)$. Given the way the parameters are specified, it requires us to compute $p(q_1, q_2, q_3 = s_2)$ and sum up for all values of q_1 and q_2 (3 x 3 = 9 addenda in total).

We have $p(q_1, q_2, q_3 = s_2) = p(q_3 = s_2 \mid q_2)\, p(q_2 \mid q_1)\, p(q_1)$.

We need an assumption for $p(q_1)$; let's make it {0.2, 0.1, 0.7}.

With a bit of patience, you can compute all 9 products of 3 factors each and obtain the desired value (my result, hopefully correct, is 0.229)

examples courtesy of Gutierrez-Osuna; and Rabiner, Juang

Estimation

- We aim to estimate the Markov chain's parameters, A and π, from one or more observed sequences of states
- Adopting MLE, we need to write the log-likelihood function, and differentiate and equate to zero for each a_{ij}, i,j=1..N, and π_i, i=1..N
- Log-likelihood function for one sequence:

$$p(Q \mid A, \pi) = p(q_1 \mid \pi) \prod_{t=2}^{T} p(q_t \mid q_{t-1}, A)$$

$$\rightarrow \quad \ln p(Q \mid A, \pi) = \ln p(q_1 \mid \pi) + \sum_{t=2}^{T} \ln p(q_t \mid q_{t-1}, A)$$

Estimation

- Log-likelihood function for E sequences, $Q^e = \{q^e_1, \ldots q^e_t, \ldots q^e_{T_e}\}$, assumed independent of each other:

$$\ln \prod_{e=1}^{E} p\left(Q^e \mid A, \pi\right) = \sum_{e=1}^{E} \ln p\left(Q^e \mid A, \pi\right) =$$

$$= \sum_{e=1}^{E} \ln p\left(q_1^e \mid \pi\right) + \sum_{e=1}^{E} \sum_{t=2}^{T_e} \ln p\left(q_t^e \mid q_{t-1}^e, A\right)$$

- When maximising the above function in the a_{ij} and π_i, one must satisfy constraints $\sum_{j=1..N} a_{ij} = 1$, $\sum_{i=1..N} \pi_i = 1$ from the axiom of total probability (constrained maximisation)
- This maximisation is direct and easy

Estimation

- Let us express the number of the occurrences of each parameter in the sequences:

$$n_i = \sum_{e=1}^{E} \delta\left(q_1^e = s_i\right)$$

$$n_{ij} = \sum_{e=1}^{E} \sum_{t=2}^{T_e} \delta\left(q_t^e = s_j, q_{t-1}^e = s_i\right)$$

- One can easily see that the log-likelihood can also be expressed in terms of n_i, n_{ij}:

$$\sum_{e=1}^{E} \ln p\left(Q^e \mid A, \pi\right) = \sum_{i=1}^{N} n_i \ln \pi_i + \sum_{i=1}^{N} \sum_{j=1}^{N} n_{ij} \ln a_{ij}$$

Estimation

- To satisfy the constraint, we need to build a Lagrangian equation. Example for π_i (λ is the Lagrangian multiplier):

$$\frac{\partial}{\partial \pi_i}\left[\sum_{i=1}^{N} n_i \ln \pi_i + \lambda\left(\sum_{i=1}^{N} \pi_i - 1\right)\right] = \frac{n_i}{\pi_i} + \lambda = 0$$

$$\rightarrow \lambda\pi_i = -n_i \rightarrow \lambda\sum_{i=1}^{N} \pi_i = -\sum_{i=1}^{N} n_i \rightarrow \lambda = -E$$

$$\rightarrow \pi_i = \frac{n_i}{E}$$

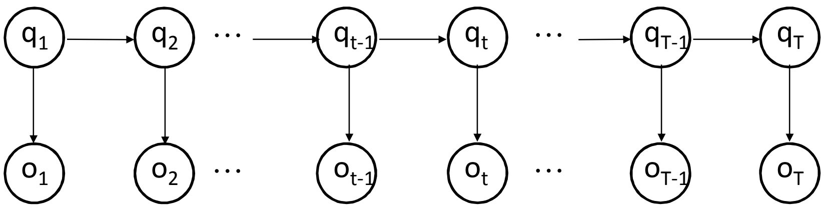
- The above is nothing else than the MLE for a discrete distribution; the solution for the a_{ij} are analogous

Hidden Markov model

- In many cases of interest, the state of a system cannot be directly observed; rather, inferred through measurements of other variables, known as *observations* (aka *measurements, emissions, outputs)*
- This implies that the state of a system at any given time can be treated as a *hidden* r.v. and observations as samples
- A *hidden Markov model* (HMM) is a probabilistic model for a sequence of observations, $O = \{o_1,\ldots,o_t,\ldots,o_T\}$ and the corresponding sequence of hidden states, $Q = \{q_1,\ldots,q_t,\ldots,q_T\}$
- The HMM models the joint probability of Q and O, p(Q,O)

HMM: graphical model

- Graphical model for the hidden Markov model:



- Each q_t, t = 1...T, takes value in a finite set of N symbols, S = $\{s_1 \dots s_N\}$
- Each o_t, t = 1...T, takes value in a finite set of M symbols, V = $\{v_1 \dots v_M\}$. Later, we'll extend the model to continuous observations

Hidden Markov model

- Fundamental hypotheses of the HMM:

1. Markov state transitions: q_t, given q_{t-1}, is independent of all the other **previous** variables:

$$p(q_t \mid q_{t-1}, o_{t-1}, .., q_1, o_1) = p(q_t \mid q_{t-1})$$

2. Independence of each observation given its state: o_t, given q_t, is independent of **all the other variables**, past and future:

$$p(o_t \mid q_T, o_T, ..q_{t+1}, o_{t+1}, q_t, q_{t-1}, o_{t-1} .., q_1, o_1) = p(o_t \mid q_t)$$

Generative model

- Joint probability of Q and O:

$$p(O,Q) = p(o_1,\dots,o_t,\dots,o_T,q_1,\dots,q_t,\dots,q_T) =$$
$$= p(o_T \mid q_T)p(q_T \mid q_{T-1})\dots p(o_t \mid q_t)p(q_t \mid q_{t-1})\dots =$$
$$= p(q_1)\left(\prod_{t=2}^{T} p(q_t \mid q_{t-1})\right)\left(\prod_{t=1}^{T} p(o_t \mid q_t)\right)$$

- Marginal probability of O:

$$p(O) = \sum_{Q} p(O,Q)$$

HMM parameters

- A: state transition probabilities: N x N matrix

$$a_{ij} = p(q_t = s_j \mid q_{t-1} = s_i)$$

- B: observation probabilities: N x M matrix

$$b_j(k) = p(o_t = v_k \mid q_t = s_j)$$

- π: initial state probabilities: N x 1 vector

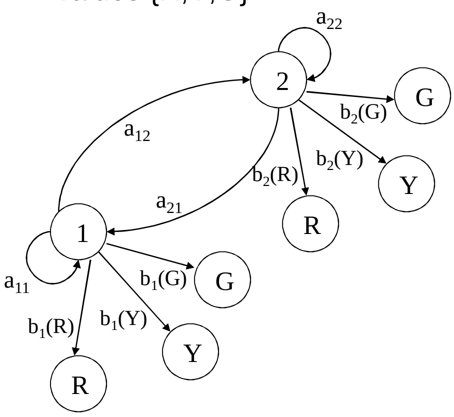
$$\pi_i = p(q_1 = s_i)$$

- Overall:

$$\lambda = \{A,B,\pi\}$$

Example

- A case with discrete outputs, 2 states {1,2} and 3 output values {R,Y,G}



	1	2
1	0.95	0.05
2	0.20	0.80

State transition matrix

	R	Y	G
1	0.45	0.10	0.45
2	0.20	0.10	0.70

Observation matrix

1	1.00
2	0.00

Initial state vector

 23

Example

- The previous example could be that of a traffic light operating in two possible modes, ‘normal’ (state 1) and ‘rush-hour’ (state 2):
  - during normal mode, the average duration of the green light is the same as the red one
  - in rush-hour mode, the green light lasts more than the red
  - we don’t know when the traffic light is in whichever mode; we can just observe its light; say, we sample it every 15 minutes
  - as the rush-hour time is shorter, we made up a_{22} smaller than a_{11}

“Why am I always coming from a side street during the rush hour?”

 24

Fundamental problems

- There are various “canonical” problems about HMM, each with well-worked solutions:
  1. **Evaluation**: given *O* and λ, compute likelihood $p(O|\lambda)$
  2. **Inference**: given *O* and λ, compute $p(Q|O,\lambda)$
  3. **Decoding**: *g*iven *O* and λ, find the “best” state sequence as

$$Q^* = \arg\max_{Q} p(Q \mid O, \lambda)$$

  4. **MLE, or learning**: given a set of training sequences, $\{O^e\}$, e = 1…E, find λ maximising

$$\lambda^* = \arg\max_{\lambda} \left(\prod_{e=1}^{E} p(O^e \mid \lambda) \right)$$

 25

Comments

- **Evaluation**: sequence O is our sample, and we want to evaluate its probability in the model, $p(O|\lambda)$ (O,λ are the equivalent of what we called x,θ in “static” data)
- **Inference**: given a sequence O, we want to know how likely is a certain sequence of states, Q. States may have a physical interpretation, like classes, and we wish to estimate their probability
- **Decoding**: given a sequence O, we want to know which is the **most likely** sequence of states, Q (equivalent to *sequential classification*)
- **Maximum likelihood estimation** or **learning** is the usual approach for learning the model’s parameters from a set of samples. NB: the states, Q, are unsupervised!

 26

Evaluation

- Target: p(O|λ), likelihood of O given λ
- It is obtained from p(O,Q|λ) by marginalising Q:

$$p(O\,|\,\lambda) = \sum_{\forall Q = q_1 q_2 \ldots q_T} p(O,Q\,|\,\lambda) = \sum_{\forall Q} p(O\,|\,Q,\lambda)\, p(Q\,|\,\lambda)$$

$$p(O\,|\,Q,\lambda) = \prod_{i=1}^{T} p(o_t\,|\,q_t,\lambda) = b_{q_1}(o_1)\, b_{q_2}(o_2) \ldots b_{q_T}(o_T)$$

$$p(Q\,|\,\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}$$

$$\rightarrow p(O\,|\,\lambda) = \sum_{\forall Q = q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(o_1)\, a_{q_1 q_2} b_{q_2}(o_2)\, a_{q_2 q_3} \ldots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

a sum of N^T addenda, each composed of 2T factors

Evaluation

- Unfortunately, there are N^T different possible values for Q!
  - for instance, with N = 5 and T = 1000, $N^T = 10^{699}$. Can you sense how large is this number??
- This makes naïve evaluation impractical. Luckily, the complexity can be reduced from exponential to linear in T
- The algorithm is known as the *forward procedure*. A *backward* and a *forward-backward* algorithms of equivalent computational cost are also possible

Forward procedure

- Let us first introduce an auxiliary quantity, $\alpha_t(j)$:

$$\alpha_t(j) = p(o_t, \ldots, o_1, q_t = s_j \mid \lambda)$$

- Initial step: $$\alpha_1(j) = \pi_j b_j(o_1)$$

- Generic step:

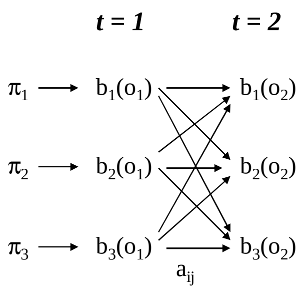
$$\alpha_t(j) = \sum_{i=1}^{N} \left(\alpha_{t-1}(i) a_{ij}\right) b_j(o_t) \quad t = 2 \ldots T,\ j = 1 \ldots N$$

  - at every time step, the number of partial products in $\alpha_t(j)$ increases by N and their length increases by 1

- Final step: $$p(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Example

- An example with N = 3:



$$\boldsymbol{\alpha_2(1)} = \pi_1 b_1(o_1) a_{11} b_1(o_2) + \pi_2 b_2(o_1) a_{21} b_1(o_2) + \pi_3 b_3(o_1) a_{31} b_1(o_2)$$

…

…

- $\alpha_1(i)$ contains 1 addendum, $\alpha_2(i)$ contains N^{2-1} addenda, $\alpha_t(i)$ contains N^{t-1} addenda, $\alpha_T(i)$ contains N^{T-1} addenda; the final sum over $\alpha_T(i)$ contains the desired N^T addenda

Backward procedure

- A "backward" auxiliary quantity, $\beta_t(i)$:

$$\beta_t(i) = p(o_{t+1},\ldots,o_T,|\, q_t = s_i,\lambda)$$

- Initial step:

$$\beta_T(i) = 1$$

- Generic step:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j) \quad t = T-1\ldots 1, i = 1\ldots N$$

- Final step:

$$p(O\,|\,\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1)\beta_1(j)$$

Forward-Backward

- Please notice that (proof omitted):

$$p(O,q_t) = p(o_1,\ldots,o_t,q_t)p(o_{t+1},\ldots,o_T,|\,q_t) = \alpha_t\beta_t$$

- The following therefore applies (just marginalise q_t):

$$p(O\,|\,\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$$

- The forward and backward procedures are just particular cases: $\Sigma_i\ \alpha_T(i)\ \beta_T(i)$ and $\Sigma_i\ \alpha_1(i)\ \beta_1(i)$, respectively

Inference

- The **inference** in HMM aims to determine p(Q|O) (or its marginals for only some of the states, like $p(q_t|O)$)
- Given the forward-backward formula, the inference comes easy:

$$p(Q\,|\,O)=\frac{p(Q,O)}{p(O)}$$

$$p(q_t\,|\,O)=\frac{p(q_t,O)}{p(O)}=\frac{\alpha_t\beta_t}{p(O)}$$

Decoding

- **Decoding** in HMM aims to find the “best” sequence of states. If no particular loss function is given, the MAP rule applies:

$$Q^* = \arg\max_Q \; p(Q\,|\,O)$$

- One could try to evaluate the above directly, but it would require N^T evaluations
- A much more efficient algorithm is required. The solution is offered by the **Viterbi algorithm** (which belongs to the more general class of the max-product algorithms) . The computational cost becomes linear in T

Decoding

- Remember that

$$\arg\max_{Q} p(Q \mid O) \neq$$
$$\left[\arg\max_{q_1} p(q_1 \mid O);..\arg\max_{q_T} p(q_T \mid O)\right]$$

  the optimal state sequence is not the sequence of the optimal individual states

- Please also note that argmax_Q p(Q,O) = argmax_Q p(Q|O) since p(O) plays no role in the maximisation
- Given the generative model of HMM, argmax_Q p(Q,O) is simpler to compute

 35

The Viterbi algorithm

- Let us first define an auxiliary quantity, $\delta_t(j)$:

$$\delta_t(j) = \max_{q_1, q_2 \ldots q_{t-1}} p\left(q_1, q_2, \ldots, q_{t-1}, q_t = s_j, o_1, o_2, \ldots, o_t \mid \lambda\right)$$

  that is the highest probability of any one path of the first t-1 states ending in $q_t = s_j$, given the first t observations

- We also introduce state:

$$\psi_t(j) = \arg\max_{i=1..N} \left(\delta_{t-1}(i) a_{ij}\right)$$

  that is the most likely value of q_{t-1} leading to $q_t = s_j$

 36

The Viterbi algorithm

- Like for the evaluation algorithm, we progress partial results. However, here we progress partial maxima rather than partial sums, for the same $O(N^2T)$ complexity
- We apply a sequence of iterative steps ending with the computation of:

$$q_T^* = \underset{i=1..N}{\arg\max}\, \delta_T(i)$$

- From there, we "roll back" for the other states using vector Ψ

The Viterbi algorithm

- Initial step: $\delta_1(j) = \pi_j b_j(o_1) \quad j = 1 \ldots N$

  $\psi_1(j) = N / A$

- Generic step: $\delta_t(j) = \max_{i=1..N}\left(\delta_{t-1}(i) a_{ij}\right) b_j(o_t) \quad j = 1 \ldots N, t = 2 \ldots T$

  $\psi_t(j) = \underset{i=1..N}{\arg\max}\left(\delta_{t-1}(i) a_{ij}\right)$

- Final step: $p_{q_1^*, \ldots, q_T^*} = \max_{i=1..N}\left(\delta_T(i)\right)$

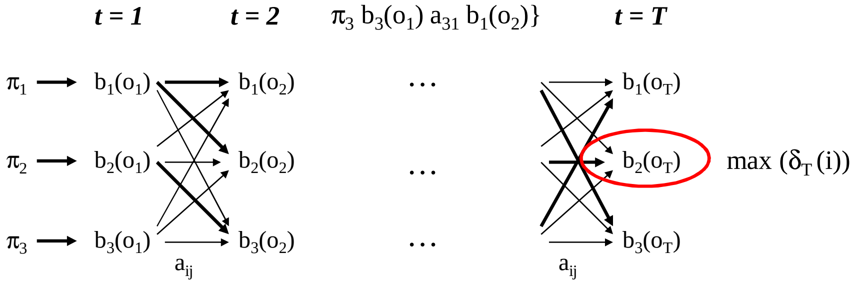
  $q_T^* = \underset{i=1..N}{\arg\max}\left(\delta_T(i)\right)$

  $q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \qquad t = (\mathrm{T}-1), \ldots, 1$

Example

- An example with N = 3:

$$\delta_2(1) = \max \{\pi_1 b_1(o_1)\, a_{11}\, b_1(o_2),\ \pi_2 b_2(o_1)\, a_{21}\, b_1(o_2),\ \pi_3 b_3(o_1)\, a_{31}\, b_1(o_2)\}$$



Learning: MLE

- We learn the model, λ, with maximum likelihood
- As data, we could even use one single sequence, O, as in:

$$\lambda = \arg\max_{\lambda} \left(p(O \mid \lambda) \right)$$

- Of course, the estimate should be more general if we use multiple, independent sequences, O^e, e=1…E, as in:

$$\lambda = \arg\max_{\lambda} \left(\prod_{e=1}^{E} p(O^e \mid \lambda) \right)$$

- In the following, we present in detail the solution with one sequence, and then extend the solution to the multiple-sequence case

Estimation

- The Baum-Welch algorithm is an EM algorithm for estimating λ given one or more observation sequences
- A full description of the Baum-Welch algorithm is given in [Bilmes98] in two ways:
  - based on the forward & backward procedures (original algorithm, disclosed in 1974)
  - based on the EM algorithm (1977)
- Both ways lead to an equivalent solution; in the following we show the iterative re-estimation formulas for A, B, and π

The Q function for HMM

$$Q\left(\lambda,\lambda^{old}\right)=\sum_{Q} ln\ p(Q,O\mid\lambda)p\left(Q\mid O,\lambda^{old}\right)$$

$$Q\left(\lambda,\lambda^{old}\right)=\sum_{q_1=1}^{N}\cdots\sum_{q_T=1}^{N} ln\ p(Q,O\mid\lambda)p\left(Q\mid O,\lambda^{old}\right)=$$

$$=\sum_{q_1=1}^{N}\cdots\sum_{q_T=1}^{N} ln\left(\pi_{q_1}\prod_{t=1}^{T-1}a_{q_t q_{t+1}}\prod_{t=1}^{T}b_{q_t}(o_t)\right)p\left(Q\mid O,\lambda^{old}\right)=$$

$$=\sum_{q_1=1}^{N}\cdots\sum_{q_T=1}^{N}\left(ln\,\pi_{q_1}+\sum_{t=1}^{T-1}ln\,a_{q_t q_{t+1}}\sum_{t=1}^{T}ln\,b_{q_t}(o_t)\right)p\left(Q\mid O,\lambda^{old}\right)$$

The Q function for HMM

- Looking at the terms under logarithm, it is clear that they depend on only a subset of the states; when multiplied by p(Q,O|λ) and integrated, the dependence on some states disappears. Example:

$$\sum_{q_1=1}^{N}\cdots\sum_{q_T=1}^{N} ln\,\pi_{q_1}\, p\left(Q\,|\,O,\lambda^{old}\right)=\sum_{q_1=1}^{N} ln\,\pi_{q_1}\, p\left(q_1\,|\,O,\lambda^{old}\right)$$

- Moreover, the HMM parameters do not depend on the time. Therefore:

$$\sum_{q_1=1}^{N}\cdots\sum_{q_T=1}^{N}\sum_{t=1}^{T-1}\ln a_{q_t q_{t+1}}\, p\left(q_t, q_{t+1}\,|\,O,\lambda^{old}\right)=$$

$$=\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\ln a_{ij}\, p\left(q_t=s_i, q_{t+1}=s_j\,|\,O,\lambda^{old}\right)$$

The Q function for HMM

- Overall:

$$Q\left(\lambda,\lambda^{old}\right)=$$

$$=\sum_{i=1}^{N} ln\,\pi_i\, p\left(q_1=s_i\,|\,O,\lambda^{old}\right)+$$

$$+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1} ln\,a_{ij}\, p\left(q_t=s_i, q_{t+1}=s_j\,|\,O,\lambda^{old}\right)+$$

$$+\sum_{i=1}^{N}\sum_{t=1}^{T-1} ln\,b_i\left(o_t=v_k\right)p\left(q_t=s_i\,|\,O,\lambda^{old}\right)$$

- The maximisation requires to differentiate Q in each of the π_i, a_{ij}, $b_i(k)$, equate to 0 and find the solutions. Constraints need to be added to make them proper probabilities

The Q function for HMM

- Please note that the same maxima are returned by the “more natural” (and efficient):

$$Q'\left(\lambda,\lambda^{old}\right)=Q\left(\lambda,\lambda^{old}\right)p\left(O\mid\lambda^{old}\right)=\sum_{Q} ln\ p(Q,O\mid\lambda)p\left(Q,O\mid\lambda^{old}\right)$$

- Given that in HMM p(Q|O) is obtained as p(Q,O)/p(O), using the above avoids a number of unnecessary normalisations. Yet, to avoid confusion, in the following we use the standard definition of $Q(\lambda,\lambda^{old})$

 45

Responsibilities

- The estimation requires defining responsibilities:

$$\gamma_t(i)=p(q_t=s_i\mid O,\lambda)$$

which is the probability of being in state s_i at time t given sequence O, and:

$$\xi_t(i,j)=p\left(q_t=s_i,q_{t+1}=s_j\mid O,\lambda\right)$$

which is the probability of being in state s_i at time t and in state s_j at time t+1 given sequence O

- $\gamma_t(i)$ and $\xi_t(i,j)$ can be updated at every iteration based on $\alpha_t(i)$ and $\beta_t(i)$ [Bilmes98]. This is the E step

 46

Re-estimation formulas

- π: $$\pi_i = \gamma_1(i)$$
- A: $$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$
- B: $$b_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t = v_k)\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Re-estimation formulas for multiple training sequences

- π: $$\pi_i = \frac{1}{E}\sum_{e=1}^{E} \gamma_1^e(i)$$
- A: $$a_{ij} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e-1} \xi_t^e(i,j)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e-1} \gamma_t^e(i)}$$
- B: $$b_i(k) = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \delta(o_t^e = v_k)\gamma_t^e(i)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e} \gamma_t^e(i)}$$

Continuous observations

- In many cases of interest, the observations are continuous r.v. Probabilities p(o|q) for the discrete case are replaced by densities p(o|q)
- Let us assume that each p(o|q) is Gaussian. B becomes a vector of N mean-covariance pairs:

$$B = \{\mu_i, \Sigma_i\}, \quad i = 1 \ldots N$$

- NB: an HMM is very much like a mixture distribution, with the difference that the probability of each component depends on what component was drawn in the previous step of the sequence (p(z) is replaced by $p(q_t|q_{t-1})$)

Continuous observations: Gaussian

- The re-estimation formulas are similar to those for conventional Gaussians; only, this time, each sample belongs to each component by a fractional membership:

$$\mu_i^{new} = \frac{\sum_{t=1}^{T} o_t \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \gamma_t(i) := p(q_t = i \mid O, \theta^{old})$$

$$\Sigma_i^{new} = \frac{\sum_{t=1}^{T} (o_t - \mu_i^{new})(o_t - \mu_i^{new})^T \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Continuous observations

- Another typical model for continuous observations is the GMM; thus, we have one GMM per state (N GMMs in total):

$$B = \{\alpha_{il}, \mu_{il}, \Sigma_{il}\}, \quad i = 1 \ldots N, l = 1 \ldots M$$

- This time, there are two fractional memberships:
  - the membership of sample o_t in state s_i, given by $\gamma_t(i)$
  - the membership of such a fractional sample in the l-th component for state s_i, $p_i(l|o_t)$

Continuous observations: GMM

$$\alpha_{il}^{new} = \frac{\sum_{t=1}^{T} p_i(l \mid o_t, \theta^{old}) \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \leftarrow p(l, i \mid O, \theta^{old}) = \\ = p(l \mid i, o_t, \theta^{old}) p(i \mid O, \theta^{old})$$

$$\mu_{il}^{new} = \frac{\sum_{t=1}^{T} o_t \, p_i(l \mid o_t, \theta^{old}) \gamma_t(i)}{\sum_{t=1}^{T} p_i(l \mid o_t, \theta^{old}) \gamma_t(i)}$$

$$\Sigma_{il}^{new} = \frac{\sum_{t=1}^{T} (o_t - \mu_{il}^{new})(o_t - \mu_{il}^{new})^T p_i(l \mid o_t, \theta^{old}) \gamma_t(i)}{\sum_{t=1}^{T} p_i(l \mid o_t, \theta^{old}) \gamma_t(i)}$$

Supervised learning

- The learning we have presented in the previous slides is unsupervised, i.e. the ground-truth states are not known during training
- In many applications, the states can be assumed known during training. The MLE objective becomes:

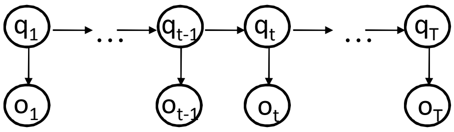
$$\lambda^* = \arg\max_{\lambda} \left(\prod_{e=1}^{E} p\left(O^e, Q^e \mid \lambda\right) \right)$$

- What changes compared to the unsupervised case?

Limitations of sequential evaluation

- In p(O,Q| λ), it is enough that one factor becomes zero for the whole product to nullify
- In the case of continuous observations, density $b_{q_t}(o_t)$ may go to zero for an unlucky observation, o_t, that scores low in the emissions of all states (an outlier). This would lead to a zero probability for the whole sequence. This problem is typically exacerbated in high dimensions
- Heavy-tailed densities such as the Student's *t* can mollify this issue

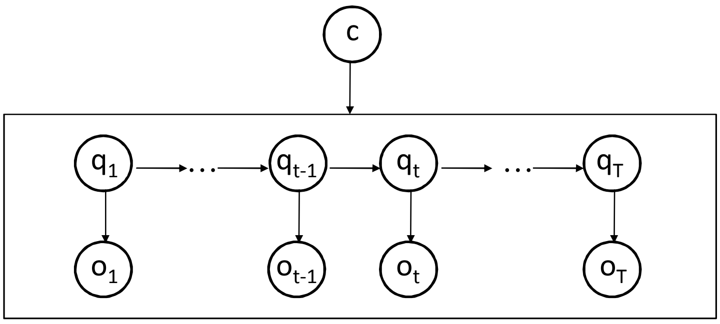
Sampling an HMM



- An HMM is a generative model and as such **it can be sampled**
- $p(Q)$ can be sampled from: $p(q_1)$, $p(q_2|q_1)$,…= π_{q_1}, $a_{q_2q_1}$,…
- $p(O|Q)$ can be sampled by fixing Q to a value and sampling from: $p(o_1|q_1)$, $p(o_2|q_2)$,…= $b_{q_1}(o_1)$, $b_{q_2}(o_2)$,…
- $p(O,Q)$ can be sampled from ancestral sampling of the above: first $p(Q)$, then $p(O|Q)$ conditioned on the samples from $p(Q)$
- $p(O)$ can be obtained from the above by simply dropping the Q component of the samples
- Less trivial, left as exercise: $p(Q|O)$. Remember that $p(Q|O) \propto p(O,Q)$, use the Markov property and the backward formula

Assigning a data sequence to a class

- Given a data sequence, O, we want to assign it to the best class, c*, out of C classes
- The graphical model becomes of the type:



Data sequence classification

- By using one HMM per class, p(Q,O|c), and Bayes' rule for classification:

$$c^* = \arg\max_c p(c \mid O) = \arg\max_c \left(p(c,O) = p(O \mid c)p(c)\right)$$

$$p(O \mid c) = \sum_Q p(Q,O \mid c)$$

- Term p(O|c) is the probability of the given sequence, O, in the HMM of class c. It can be easily computed with the forward-backward algorithm

Some HMM variations

- HMM with explicit state duration modelling (hidden semi-Markov model)
- Coupled HMM
- Hierarchical HMM
- Layered HMM
- Factorial HMM
- Input-Output HMM
- Hierarchical Dirichlet Process HMM
- Dynamic Bayesian networks (DBNs) and generative graphical models in general

Software

- K. Murphy, Software for graphical models: a review, ISBA Bulletin, 14(4), Dec. 2007:
  - contains a recent review of the most popular **software packages for graphical models**, HMM included
- The early (1998) K. Murphy, Hidden Markov Model (HMM) Toolbox for Matlab, http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
- The recent pmtk3, probabilistic modeling toolkit for Matlab/Octave, version 3, http://code.google.com/p/pmtk3/
- Matlab Statistics Toolbox™ 7.0
  - includes functions for discrete HMMs
- GHMM (from the Algorithmics group at the Max Planck Institute for Molecular Genetics), http://www.ghmm.org/
  - C library implementing efficient data structures and algorithms for basic and extended HMMs
- much more

An example with Murphy's toolbox

- This example is slightly modified from Murphy's "`dhmm_em_demo`"
- We create a model with the assumed initial probabilities, transition and observation matrices of our traffic light ("true" model)
- We sample 20 training sequences, each of length 100, from the true model
- We initialise the training of the HMM from arbitrary values; the transition probabilities between states are set to symmetric values
- We run EM, max iterations: 200

The traffic light example

```
Q = 2; % number of values for the state variables
O = 3; % number of values for the observation variables

% "true" parameters:
prior0 = [1 0];
transmat0 = [0.95 0.05; 0.20 0.80];
mk_stochastic(transmat0)
obsmat0 = [0.45  0.10 0.45; 0.20  0.10 0.70];
mk_stochastic(obsmat0)

% training data, sampled from the true model:
T = 100;
nex = 20;
data = dhmm_sample(prior0, transmat0, obsmat0, T, nex);

% prints the log-likelihood of the sample in the true model:
loglik = dhmm_logprob(data, prior0, transmat0, obsmat0)
```


The traffic light example (2)

```
% initial, arbitrary guess of parameters:
prior1 = [0.8 0.2];
transmat1 = [0.8 0.2; 0.2 0.8];
mk_stochastic(transmat1)
obsmat1 = [0.5 0.25 0.25; 1/3 1/3 1/3];
mk_stochastic(obsmat1)

% prints the log-likelihood of the samples in the initial model:
loglik = dhmm_logprob(data, prior1, transmat1, obsmat1)

% learned parameters using EM:
[LL, prior2, transmat2, obsmat2] = dhmm_em(data, prior1, transmat1,
  obsmat1, 'max_iter', 200);

% parameters learned:
prior2
transmat2
obsmat2

% prints the log-likelihood of the samples in the learned model:
loglik = dhmm_logprob(data, prior2, transmat2, obsmat2)
```

The traffic light example: example of output

- Every run differs because of the different sampled data
- Example of learned values for one run:

```
prior2    = [0.7976  0.2024]
transmat2 = [0.7942  0.2058;  0.1923  0.8077]
obsmat2   = [0.4791  0.0975  0.4234;  0.3093  0.1289  0.5618]
```

- The estimate of the initial probabilities and the transition probabilities is rather poor; instead, the observation probabilities capture the desired asymmetry and the low frequency of the yellow case
- One can check that the samples' likelihood in the learned model can be even higher than that in the true model (!), especially if the number of samples is low. Maximum likelihood learning tends to fit the model to the samples very tightly

References

- L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. IEEE, vol. 77, no. 2, pp. 257-286, 1989
- B. H. Juang and L. R. Rabiner, "A Probabilistic Distance Measure for Hidden Markov Models," AT&T Tech Journ., Vol. 64, No. 2, pp. 391-408, February 1985
- Bilmes, J. "A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models," Tech. Rep. ICSI-TR-97-021, University of California Berkeley, 1998
- Murphy, K., Hidden Markov Model (HMM) Toolbox for Matlab, http://www.ai.mit.edu/~murphyk/Software/HMM/hmm.html