Maximum Likelihood and Bayesian Parameter Estimation

Pattern Classification by Richard O. Duda, Peter E . Hart and David G. Stork
Chapter 3 - Part 2

12 December 2018
Luciana Majercsik

Outline

Expectation Maximization

Hidden Markov Models (HMM)

Expectation-Maximization (EM)

Introduction

EM is a general method for finding the ML estimate of the parameters of a pdf when the data has missing values

Two main applications of the EM algorithm :

- When the data indeed has incomplete, missing or corrupted values as a result of a faulty observation process
- When assuming the existence of missing or hidden parameters can simplify the likelihood function, which would otherwise lead to an analytically intractable optimization problem;

Simple Example*

Let events be “grades in a class”

w_1 = Gets an A	$P(A) = \frac{1}{2}$
w_2 = Gets a B	$P(B) = \mu$
w_3 = Gets a C	$P(C) = 2\mu$
w_4 = Gets a D	$P(D) = \frac{1}{2}-3\mu$
	(Note $0 \leq \mu \leq 1/6$)

Assume we want to estimate μ from data. In a given class there were

a A’s
b B’s
c C’s
d D’s

What’s the maximum likelihood estimate of μ given a, b, c, d ?

* http://www.cs.cmu.edu/~cga/ai-course/gmm.pdf

Trivial Statistics

As P(A) = ½   P(B) = μ   P(C) = 2μ   P(D) = ½-3μ we have:

$$P(a,b,c,d\mid\mu)=K\left(\frac{1}{2}\right)^{a}\mu^{b}(2\mu)^{c}\left(\frac{1}{2}-3\mu\right)^{d}$$

Which implies:

$$\ln P(a,b,c,d\mid\mu)=\ln K+a\log\left(\frac{1}{2}\right)+b\ln\mu+c\ln(2\mu)+d\ln\left(\frac{1}{2}-3\mu\right)$$

We obtain the solution that maximizes the log likelihood function by setting:

$$\frac{\partial\ln \mathrm{P}}{\partial\mu}=0\Rightarrow\frac{b}{\mu}+\frac{2c}{2\mu}-\frac{3d}{1/2-3\mu}=0\quad\Rightarrow\quad\mu=\frac{b+c}{6\left(b+c+d\right)}$$

If in the class the grade situation is given by:

A	B	C	D
14	6	9	10

Then $\mu_{\max} = \frac{1}{10}$

Same Problem - Hidden Information

Someone tells us that:

- Number of High grades (A’s + B’s) = *h*
- Number of C’s = *c*
- Number of D’s = *d*

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

What is the maximum likelihood estimate of μ now?

Same Problem - Hidden Information

We can answer this question circularly:

If we know the value of μ we could compute the expected value of *a* and *b:*

$$a = \frac{\frac{1}{2}}{\frac{1}{2} + \mu} h \qquad b = \frac{\mu}{\frac{1}{2} + \mu} h$$

EXPECTATION

If we know the expected values of *a* and *b* we could compute the maximum likelihood value of μ (as before):

$$\mu = \frac{b + c}{6(b + c + d)}$$

MAXIMIZATION

We begin with a guess for μ . We iterate between **EXPECTATION** and **MAXIMALIZATION** to improve our estimates of μ and *a* and *b*.

Let μ(t) be the estimate of μ on the t'th iteration and b(t) the estimate of *b* on t'th iteration. Then, with:

$$\mu(0) = \text{ initial guess}$$

$$b(t) = \frac{\mu(\mathrm{t})h}{\frac{1}{2} + \mu(t)} = \mathrm{E}\left[b \mid \mu(t)\right]$$

E Step

$$\mu(t+1) = \frac{b\left(t\right) + c}{6\left(b\left(t\right) + c + d\right)}$$

M step

Here μ(t+1) being the maximum likelihood estimate of μ, given b(t).

We have to continue iterating until it converges. Good news: Converging to local optimum is assured. Bad news: the optimum is "local" .

- In our example, suppose we had h = 20, c = 10, d = 10 and μ(0) = 0. Then:

t	μ(t)	b(t)
0	0	0
1	0.0833	2.857
2	0.0937	3.158
3	0.0947	3.185
4	0.0948	3.187
5	0.0948	3.187

Mathematical form of EM

- Consider a full sample $D = \{x_1,..., x_N\}$ of points taken from a single distribution. Suppose, though, that here some features are missing; thus any sample point can be written as $x_k = \{x_{kg}, x_{kb}\}$, i.e., comprising the "good" features and the missing, or "bad" ones

- We separate these individual features (not samples) into two sets, D_g and D_b with $D = D_g \cup D_b$ being the union of such features.

- Next we form the function:

$$Q\left(\theta;\theta^i\right) = E_{D_b}\left[\ln p\left(D_g, D_b; \theta\right) \mid D_g; \theta^i\right]$$

- $Q(\theta; \theta^i)$ is a function of θ with θ^i assumed fixed. On the right hand side is the expected value of ln $p(D_g , D_b , \theta)$ with respect to the unknown data D_b, given the data D_g and the current parameter estimates θ^i. θ^i is the current (best) estimate for the full distribution

Mathematical form of EM

- **EXPECTATION STEP**

  – Find the expected value of ln p(D_g , D_b , θ) with respect to the unknown data D_b, given the data D_g and the current parameter estimate θ^i

- **MAXIMIZATION STEP**

  – Find the argument θ that maximizes the expected value $Q(\theta; \theta^i)$

$$\theta^{i+1} = \arg\max Q(\theta; \theta^i)$$

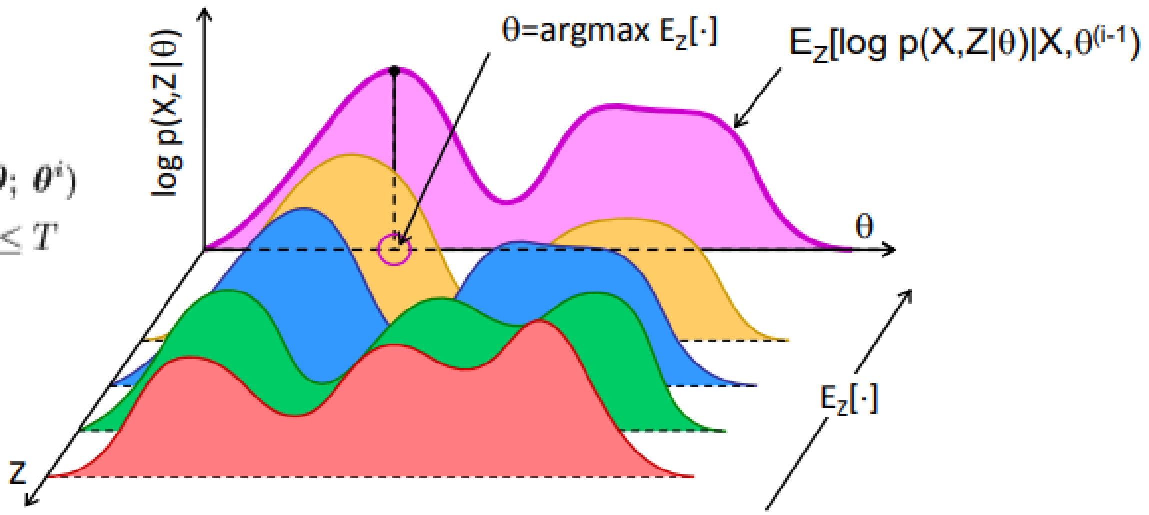
- **CONVERGENCE PROPERTIES**:

It can be shown that

1) each iteration (E+M) is guaranteed to increase the log-likelihood and
2) the EM algorithm is guaranteed to converge to a local maximum of the likelihood function

Algorithm 1 (Expectation-Maximization)

```
begin initialize $\theta^0, T, i = 0$
        do $i \leftarrow i + 1$
            E step : compute $Q(\boldsymbol{\theta};\ \boldsymbol{\theta}^i)$
            M step : $\boldsymbol{\theta}^{i+1} \leftarrow \arg\max_{\theta} Q(\boldsymbol{\theta};\ \boldsymbol{\theta}^i)$
        until $Q(\boldsymbol{\theta}^{i+1};\ \boldsymbol{\theta}^i) - Q(\boldsymbol{\theta}^i;\ \boldsymbol{\theta}^{i-1}) \leq T$
    return $\hat{\theta} \leftarrow \boldsymbol{\theta}^{i+1}$
end
```



Remarks:

This process iterates until no value of θ can be found that will increase $Q(\cdot,\cdot)$.

The Expectation-Maximization or EM algorithm is most useful when the optimization of $Q(\cdot,\cdot)$ is simpler than that of the likelihood function $l(\cdot)$.

Hidden Markov models (HMM)

Sub-Chapter Outline

Discrete Markov models

Introduction

So far, we have focussed primarily on sets of data points that were assumed to be independent and identically distributed (i.i.d.)

- This assumption allowed us to express the likelihood function as the product over all data points of the probability distribution evaluated at each data point.
- For many applications, however, the i.i.d. assumption of data will be a poor one. It is too restrictive to be of practical use for realistic problems

In the subsequent chapter, we consider a particularly important class of such data sets, namely those that describe sequential data

- These sets of data arise through measurement of time series (**example**: the rainfall measurements on successive days at a particular location, the daily values of a currency exchange rate, the acoustic features at successive time frames used for speech recognition).
- Sequential data can also arise in contexts other than time series (**example:** the sequence of nucleotide base pairs along a strand of DNA or the sequence of characters in an English sentence)

For convenience, we shall sometimes refer to ‘past’ and ‘future’ observations in a sequence. However, the models explored in this chapter are equally applicable to all forms of sequential data, not just temporal sequences

Poor assumptions for sequential data:

1. the independence assumption;
2. assume a general dependence of future observations on all previous observations (the complexity of such a model would grow without limit as the number of observations increases).

This leads us to consider **Markov models** in which we assume that future predictions are independent of all but the most recent observations

Consider a system described by the following process:

- At any given time, the system can be in one of N possible states
- At regular times, the system undergoes a **transition to a new state**
- **Transition** between states c**an be described probabilistically**

First of all we note that, without loss of generality, we can use the product rule to express the joint distribution for a sequence of observations in the form

$$P(\omega_1, \omega_2, \ldots, \omega_N) = \prod_{k=1}^{N} P(\omega_k \mid \omega_1, \ldots, \omega_{k-1})$$

Assumption: We will also assume that the transition probability between any two states is **independent of time.**

Markov property:

Assumption: the state of the system depends only on its immediate past. This is known as a first-order Markov Process.

$$\begin{tikzcd}\omega_1 \arrow[r] & \omega_2 \arrow[r] & \omega_3 \arrow[r] & \omega_4 \arrow[r] & {}\end{tikzcd}$$

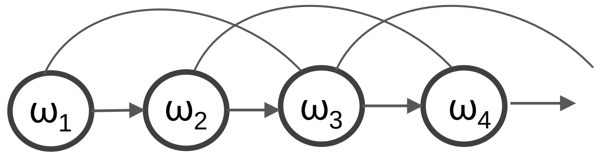
In this case, the joint distribution for a sequence of observations has the particular form

$$P(\omega_1, \omega_2, \ldots, \omega_N) = P(\omega_1) \prod_{k=2}^{N} P(\omega_k \mid \omega_{k-1})$$

Although this is more general than the independence model, it is still very restrictive.

For many sequential observations, we anticipate that the trends in the data over several successive observations will provide important information in predicting the next value.

-One way to allow earlier observations to have an influence is to move to higher-order Markov chains. If we allow the predictions to depend also on the previous-but-one value, we obtain a second-order Markov chain, represented by the graph below



We can similarly consider extensions to an M^{th} order Markov chain in which the conditional distribution for a particular variable depends on the previous M variables. However, we have paid a price for this increased flexibility because the number of parameters in the model is now much larger.

Elements of a first order Markov Model

Notation: if the state of the model at time t is ω_i then we denote this by: $\omega_i(t) \Leftrightarrow \omega(t) = \omega_i$

A first order Markov Model is characterized by the following set of parameters:

- N, the number of states in the model $\{\omega_1, \omega_2, \dots, \omega_N\}$
- The state transition probability matrix A = $\{a_{ij}\}$

$$a_{ij} = P\left(\omega_j(t+1) \mid \omega_i(t)\right), \;\; s.t. \begin{cases} a_{ij} \geq 0 \\ \sum_{j=1}^{N} a_{ij} = 1 \end{cases}$$

- If there is a prior probability on the first state $P(\omega(1) = \omega_i)$, we could include such an information in the model;

Example

Consider a simplified three-state Markov model of the weather: Any given day, the weather can be described as being :

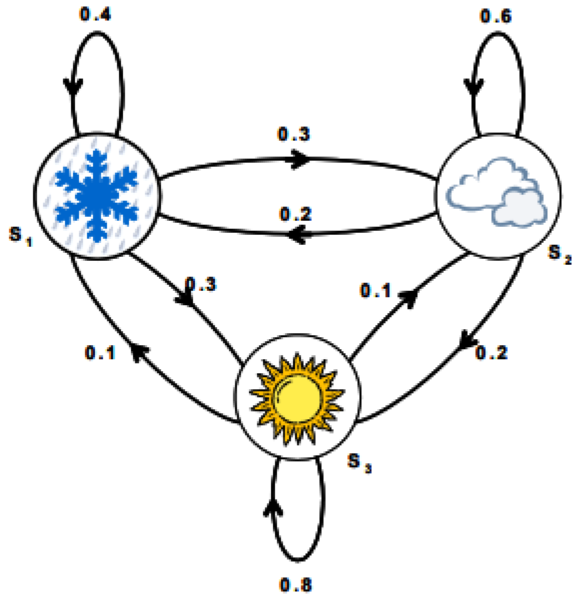
State 1: precipitation (rain or snow)

State 2: cloudy

State 3: sunny

Transitions between states are described by the transition matrix:

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$



Question:

• 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:

$$P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 \mid model) =$$
$$= P(S_3)P(S_3 \mid S_3)P(S_3 \mid S_3)P(S_1 \mid S_3)P(S_1 \mid S_1)P(S_3 \mid S_1)P(S_2 \mid S_3)P(S_3 \mid S_2) =$$
$$= \pi_3\, a_{33}\, a_{33}\, a_{31}\, a_{11}\, a_{13}\, a_{32}\, a_{23} \;=\; 1 \times 0.8 \times 0.8 \times 0.1 \times 0.4 \times 0.3 \times 0.1 \times 0.2$$

Question:

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

• Answer: $P(\underbrace{S_i, S_i, \ldots, S_i}_{T\,days}, S_{j \neq i} \mid model) =$

$$= P(S_i)\underbrace{P(S_i \mid S_i)P(S_i \mid S_i) \ldots P(S_i \mid S_i)}_{T-1} P(S_{j \neq i} \mid S_i) = a_{ii}^{T-1}(1 - a_{ii})$$

First-order Hidden Markov Models

Introduction

The Markov model assumes that each state can be uniquely associated with an observable event

- Once an observation is made, the state of the system is then trivially retrieved
- This model, however, is too restrictive to be of practical use for most realistic problems

To make the model more flexible, we will assume that the outcomes or observations of the model are a probabilistic function of each state

- Each state can produce a number of outputs according to a unique probability distribution, and each distinct output can potentially be generated at any state
- These are known a **Hidden Markov Models (HMM)**, because the state sequence is not directly observable, it can only be approximated from the sequence of observations produced by the system

Elements of an HMM

An HMM is characterized by the following set of parameters:

- N, the number of states in the model $\{\omega_1, \omega_2, ..., \omega_N\}$
- M, the number of discrete observation symbols $\{v_1, v_2, ..., v_M\}$
- $A = \{a_{ij}\}$, the state transition probability $a_{ij} = P\left(\omega_j(t+1) \mid \omega_i(t)\right)$
- $B = \{b_{jk}\}$, the observation or emission probability distribution $b_{jk} = P\left(v_k(t) \mid \omega_j(t)\right)$
- π, the initial state distribution $\pi_i = P\left(\omega(1) = \omega_i\right)$

Therefore, an HMM is specified by two scalars (N and M) and three probability distributions (A, B, and π)

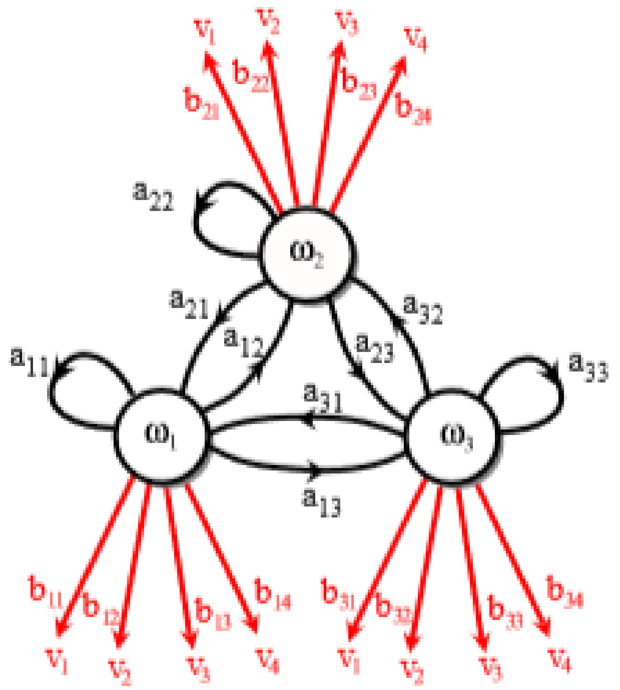
- In what follows, we will represent an HMM by the compact notation $\theta = \left(A, B, \pi\right)$

Example

Three hidden units in an HMM.

The transitions between them are shown in black while the visible states and the emission probabilities of visible states are shown in red.

This model shows all transitions as being possible; in other HMMs, some such candidate transitions are not allowed



HMM generation of observation sequences

Given a completely specified HMM θ = (A, B, π) , how can an observation sequence V = {v_1 , v_2 , v_3 , v_4 , ... } be generated?

1. Choose an initial state ω(1) according to the initial state distribution π
2. Set t = 1
3. Generate observation v_t according to the emission probability b_{jk}
4. Move to a new state ω(t+1) according to state-transition at that state a_{ij}
5. Set $t = t + 1$ and return to 3 until $t \geq T$

The coin-toss problem*

Let us consider the following scenario:

- You are placed in a room with a curtain and behind the curtain there is a person performing a coin-toss experiment
- This person selects one of several coins, and tosses it: heads (H) or tails (T)
- She tells you the outcome (H,T), but not which coin was used each time

Goal: build a probabilistic model that best explains a sequence of observations V = $\{v_1, v_2, v_3, v_4, \ldots\} = \{H, T, T, H \ldots\}$

- The coins represent the states; these are hidden because you do not know which coin was tossed each time
- The outcome of each toss represents an observation
- A "likely" sequence of coins may be inferred from the observations, but this state sequence will not be unique

If the coins are hidden, how many states should the HMM have?

* L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. of the IEEE, Vol.77, No.2, pp.257--286, 1989.

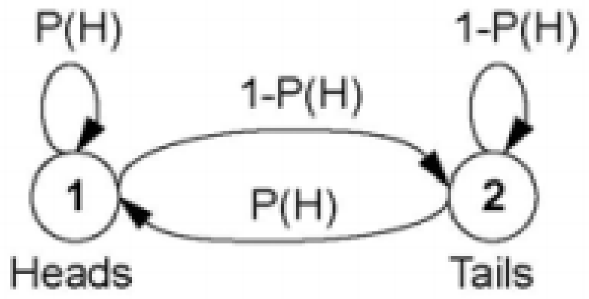
One coin model

Assumption: the person behind the curtain has only one coin

There are two states in the model, but each state is uniquely associated with either heads (state 1) or tails (state 2). Hence this model is not hidden because the observation sequence uniquely defines the state.

• We may describe the system with a model where the states are the actual observations (see figure)

• The model parameter P(H) may be found from the ratio of heads and tails



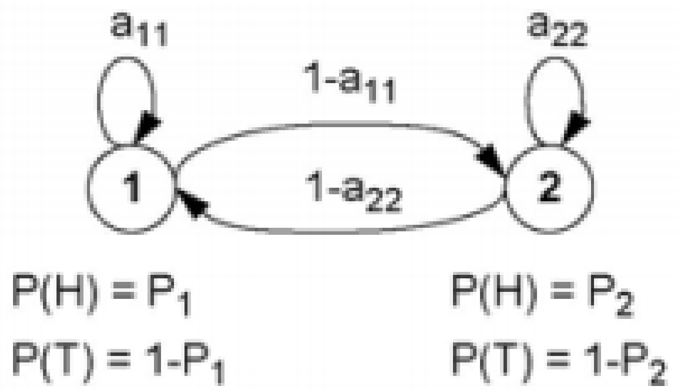
Two - coin model

Assumption: the person behind the curtain has two coins. Then we have a more sophisticated HMM.

• Each coin (state) has its own distribution of heads and tails, to model the fact that the coins may be biased

• Transitions between the two states model the random process used by the person behind the curtain to select one of the coins

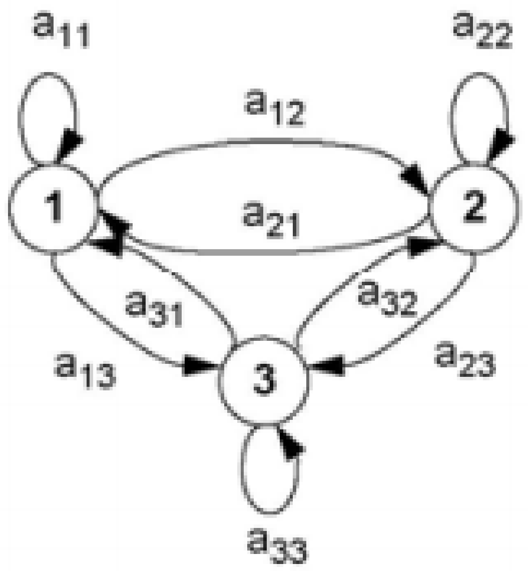
• The model has 4 free parameters



Three - coin model

Assumption: the person behind the curtain has three coins

- In this case, the model would have three separate states
- This HMM can be interpreted in a similar fashion as the two-coin model
- The model has 9 free parameters



P(H):	P_1	P_2	P_3
P(T):	$1-P_1$	$1-P_2$	$1-P_3$

1, 2 or 3 coins ?

Which of these models is best?

- Since the states are not observable, the best we can do is select the model that best explains the data (e.g., using a Maximum Likelihood criterion)

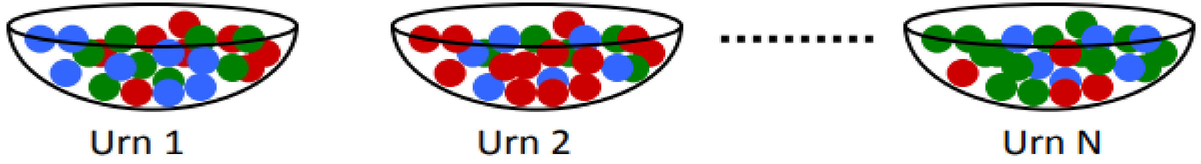
- Whether the observation sequence is long and rich enough to warrant a more complex model is a different story, though

The urn-ball problem

Let us consider the following scenario:

- You are placed in a room with a curtain and behind the curtain there are N urns, each containing a large number of balls from M different colors
- The person behind the curtain selects an urn according to an internal random process, then randomly grabs a ball from the selected urn
- She/he shows you the ball, and places it back in the urn
- This process is repeated over and over

Problem: How would you represent this experiment with an HMM? What are the states? Why are the states hidden? What are the observations?



The three central issues in HMM

The Evaluation Problem.

- Suppose we have an HMM, complete with transition probabilities a_{ij} and b_{jk}. Determine the probability (the likelihood) that a particular sequence of observations $\mathbf{V}^T$ was generated by that model.
- The solution is given by the Forward and Backward procedure

The Decoding (Optimal State Sequence) Problem.

- Suppose we have an HMM as well as a set of observations $\mathbf{V}^T$. Determine the most likely sequence of hidden states ω^T that led to those observations.
- The solution is given by the Viterbi algorithm

The three central issues in HMM

The Learning (Parameter Estimation) Problem.

- Suppose we are given the coarse structure of a model (the number of states and the number of visible states) but not the probabilities a_{ij} and b_{jk}. Given a set of training observations of visible symbols, determine these parameters. (How do we adjust the parameters of the model θ = (A, B, π) to maximize the likelihood $P(\mathbf{V}^T | \theta)$
- The solution is given by the Baum-Welch re-estimation procedure

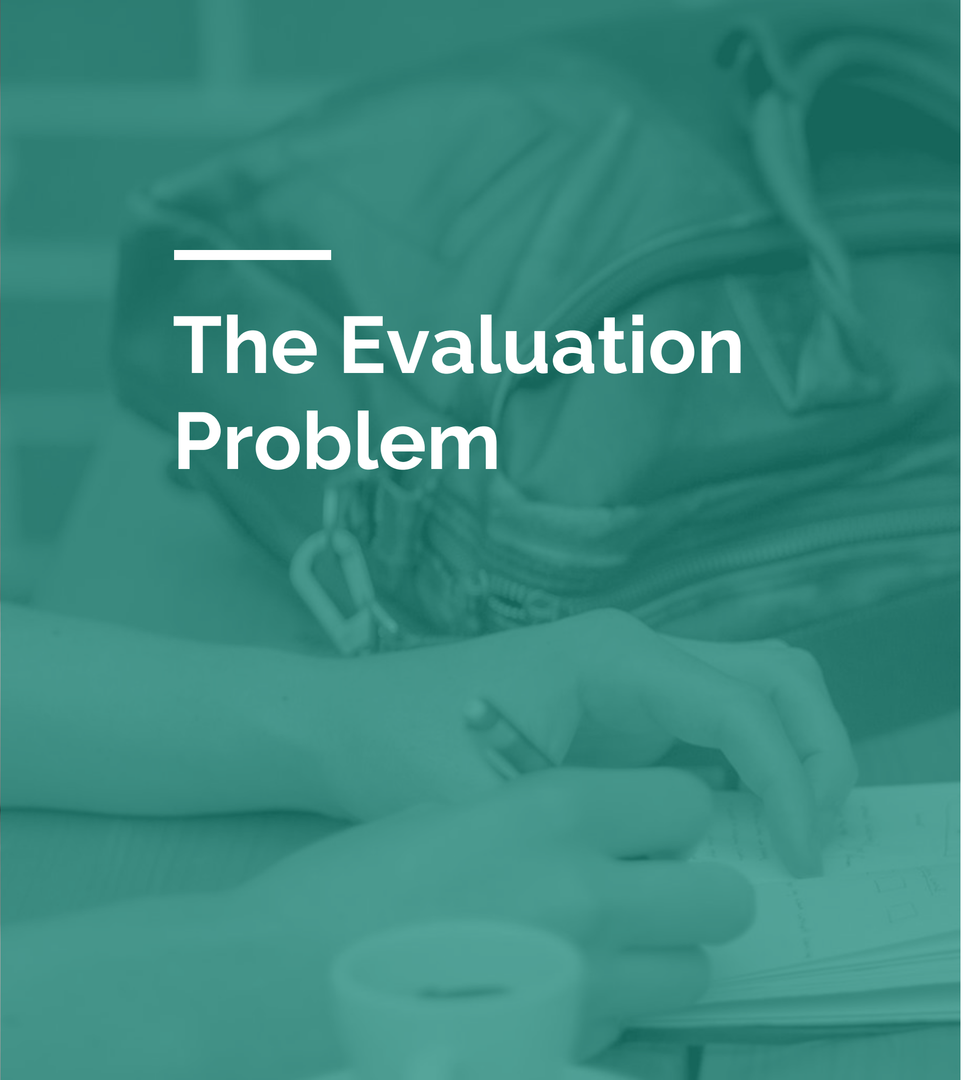
Example

To fix the ideas discussed before, we consider the case of a simple isolated word speech recognizer. For each word of a W word vocabulary, we want to design a separate N-state HMM. We represent the speech signal of a given word as a time sequence of coded spectral vectors. We assume that the coding is done using a spectral codebook with M unique spectral vectors; hence each observation is the index of the spectral vector closest (in some spectral sense) to the original speech signal. Thus, for each vocabulary word, we have a training sequence consisting of a number of repetitions of sequences of code book indices of the word (by one or more talkers).

The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate model parameters for each word model.

To develop an understanding of the physical meaning of the model states, we use the solution to Problem 2 to segment each of the word training sequences into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state. The goal here would be to make refinements on the model (e.g., more states, different codebook size, etc.) so as to improve its capability of modeling the spoken word sequences.

Finally, once the set of W HMMs has been designed and optimized and thoroughly studied, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based upon the given test observation sequence, and select the word whose model score is highest (has the highest likelihood).



The Evaluation Problem

Goal: compute the likelihood of an observation sequence $V^T = \{v_1, v_2, ..., v_T\}$ given a particular HMM model

- We can also view the problem as one of scoring how well a given model matches a given observation sequence.
- This viewpoint is extremely useful if, for example, we consider the case in which we are trying to choose among several competing models. Then the solution to the Evaluation Problem allows us to choose the model which best matches the observations.

Computation of this probability involves enumerating **every possible state sequence** and evaluating the corresponding probability

$$P(V^T) = \sum_{r=1}^{R_{\max}} P(V^T \mid \omega_r^T) P(\omega_r^T)$$

where each r indexes a particular sequence $\boldsymbol{\omega}^T_r$ = {ω(1), ω(2),..., ω(T)} of T hidden states.

Because we are dealing here with a first-order Markov process, the second factor in Eq. (1), which describes the transition probability for the hidden states, can be rewritten as:

$$P(\omega_r^T) = \prod_{t=1}^{T} P(\omega(t) \mid \omega(t-1))$$

that is, a product of the a_{ij}'s according to the hidden sequence in question. In Eq. (2), ω(T) = ω_0 is some final absorbing state, which uniquely emits the visible state v_0.

Example: In speech recognition applications, ω_0 typically represents a null state or lack of utterance, and v_0 is some symbol representing silence.

Because of our assumption that the output probabilities depend only upon the hidden state, we can write the first factor in Eq. (1) as

$$P(V^T \mid \omega_r^T) = \prod_{t=1}^{T} P(v(t) \mid \omega(t))$$

that is, a product of b_{jk}'s according to the hidden state and the corresponding visible state. We can now use Eqs. (2) & (3) to express Eq. (1) in the equivalent form:

$$P(V^T) = \sum_{r=1}^{R_{\max}} \prod_{t=1}^{T} P(v(t) \mid \omega(t)) P(\omega(t-1) \mid \omega(t))$$

Interpretation of (4): The probability that we observe the particular sequence of T visible states VT is equal to the sum over all R_{max} possible sequences of hidden states of the conditional probability that the system has made a particular transition multiplied by the probability that it then emitted the visible symbol in our target sequence.
All these informations are captured in the parameters a_{ij} and b_{kj}, and thus Eq.(4) can be evaluated directly.

Computational complexity

- With N^T possible state sequences, this approach becomes unfeasible even for small problems

  For $N = 10$ and $T = 20$, the order of computations is in the order of 10^{21}

- Fortunately, we can compute $P(\mathbf{V}^T)$ recursively, since each term $\mathbf{P(v(t)|\omega(t))P(\omega(t)|\omega(t-1))}$ involves only v(t), ω(t) and ω(t−1). This leads to a very efficient implementation known as the **Forward procedure**

The Forward Algorithm

We define: $\alpha_i(t) = P(v_1, v_2, ..., v_t, \omega_i(t) | \theta)$ which represents the probability that our HMM and is in the state ω_i at time t, having generated the first t elements of the observation sequence.

The computation of this variable can be efficiently performed by induction:

$$\alpha_i(t) = \begin{cases} 0, & t = 0 \ and \ i \neq initial \ state \\ 1, & t = 0 \ and \ i = initial \ state \\ \sum_{j=1}^{N} \alpha_j(t-1) a_{ij} b_{jk} v(t), & otherwise \end{cases}$$

where the notation $b_{jk}v(t)$ means the transition probability b_{jk} selected by the visible state emitted at time t. Thus the only non-zero contribution to the sum is for the index k which matches the visible state v(t).

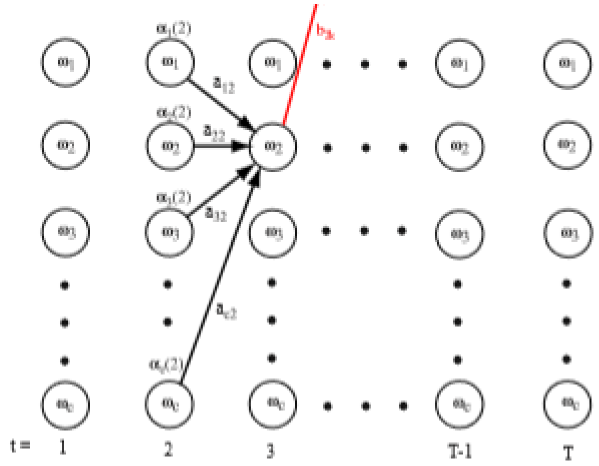
The Forward Algorithm

Algorithm 2 (HMM Forward)

```
1 initialize  $\omega(1), t = 0, a_{ij}, b_{jk}$, visible sequence $\mathbf{V}^T, \alpha(0) = 1$
2    for $t \leftarrow t + 1$
3         $\alpha_j(t) \leftarrow \sum_{i=1}^{c} \alpha_i(t-1) a_{ij} b_{jk}$
4    until $t = T$
5 return $P(\mathbf{V}^T) \leftarrow \alpha_0(T)$
6 end
```

In line 5, α_0 denotes the probability of the associated sequence ending to the known final state.

The Forward algorithm has, a computational complexity of $O(N^2T)$

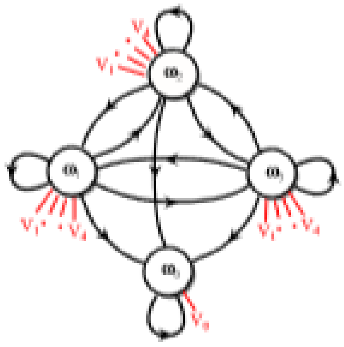


The computation of probabilities by the Forward algorithm can be visualized by means of a trellis — a sort of “unfolding” of the HMM through time.

Example

Consider an HMM with an explicit absorber state and unique null visible symbol v_0 with the following transition probabilities (where the matrix indexes begin at 0):

$$A = \{a_{ij}\} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.0 & 0.1 \end{bmatrix} \qquad B = \{b_{jk}\} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{bmatrix}$$



What is the probability it generates the particular sequence $\mathbf{V}^5 = \{v_3, v_1, v_3, v_2, v_0\}$?

Suppose we know the initial hidden state at t = 0 to be ω_1

The visible symbol at each step is shown above.

In each node is $\alpha_i(t)$—the probability the model generated the observed visible sequence up to t.

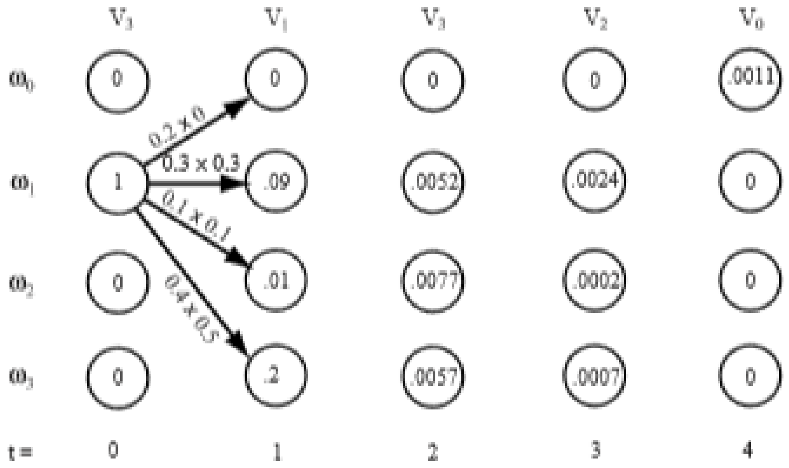
We know that the system was in hidden state ω_1 at t =0, and thus $\alpha_1(0) = 1$ and $\alpha_i(0) = 0$ for i != 1.

The arrows show the calculation of $\alpha_i(1)$. For instance, since visible state v_1 was emitted at t = 1, we have:

$\alpha_0(1) = \alpha_1(0)a_{10}b_{01} = 1[0.2\times0] = 0$ (as shown by the top arrow).

$\alpha_1(1) = \alpha_1(0)a_{11}b_{11} = 1[0.3\times0.3] = 0.09$ (the next highest arrow)

The product $a_{ij}b_{jk}$ is shown along each transition link for the step t = 0 to t = 1.



The final probability, $P(\mathbf{V}^T|\theta) = 0.0011$

The Backward Algorithm

We define analogously, the backward variable: $\beta_i(t) = P(v_{t+1}, v_{t+2}, \ldots, v_T \mid \omega_i(t), \theta)$ which represents the probability of the partial observation sequence from $t + 1$ to the end, given state ω_i at time t and model . The computation of this variable can be also efficiently performed by induction:

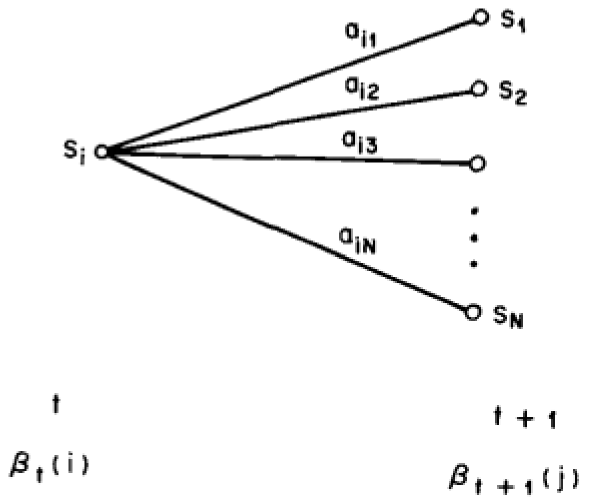
$$\beta_i(t) = \begin{cases} 1, & t = T \ \ and \ \ i = any \ \ state \\ \sum_{j=1}^{N} \beta_i(t+1) a_{ij} b_{jk} v(t+1), & otherwise \end{cases}$$

where the notation $b_{jk}v(t+1)$ means the transition probability b_{jk} selected by the visible state emitted at time t+1. Thus the only non-zero contribution to the sum is for the index k which matches the visible state v(t+1).

The Backward Algorithm

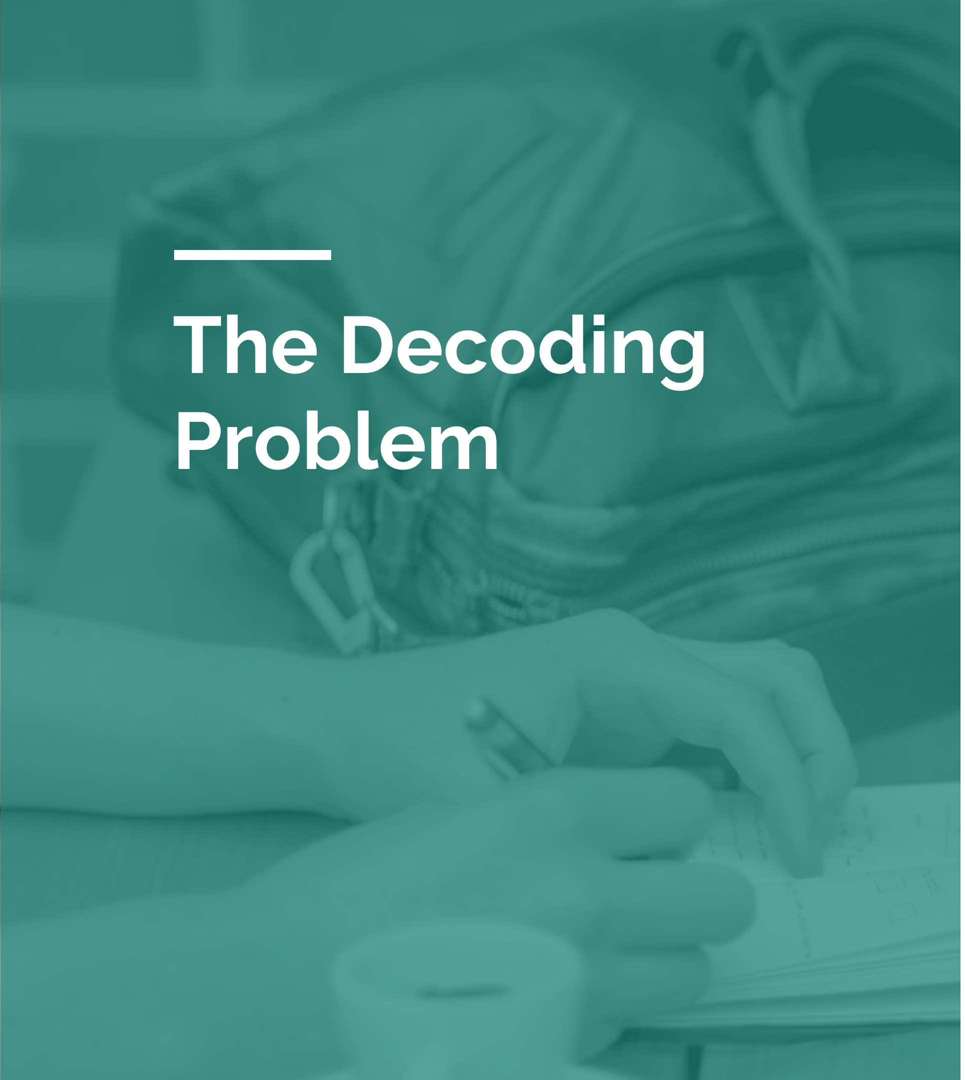
Algorithm 3 (HMM Backward)

```
1 initialize  $\omega(T), t = T, a_{ij}, b_{jk}$, visible sequence $V^T$
2    for $t \leftarrow t - 1$;
4          $\beta_j(t) \leftarrow \sum_{i=1}^{c} \beta_i(t+1) a_{ij} b_{jk} v(t+1)$
5    until $t = 1$
7 return $P(V^T) \leftarrow \beta_i(0)$ for the known initial state
8 end
```



The Backward algorithm has also a computational complexity of $O(N^2T)$

Remark: Strictly speaking, only the forward procedure would suffice to solve Problem 1. The backward part was introduced in this section since it will be used to help solve Problem 3.



The Decoding Problem

Goal: Given a sequence of visible states $V^T = \{v_1, v_2, \ldots, v_T\}$, find the most probable sequence of hidden states

- This problem is the one in which we attempt to uncover the hidden part of the model, i.e., to find the "correct" state sequence
- It should be clear that for all but the case of degenerate models, there is no "correct" state sequence to be found. Hence for practical situations, we usually use an optimality criterion to solve this problem as best as possible.

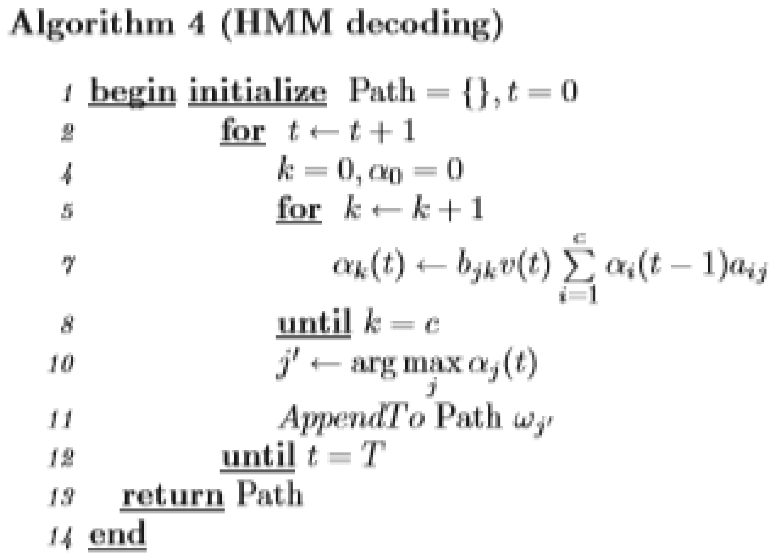
- Unlike Problem 1 for which an exact solution can be given, there are several possible ways of solving the decoding problem, namely finding the "optimal" state sequence associated with the given observation sequence.

- **Difficulty: what is the definition of the optimal state sequence**?
    - There are several possible optimality criteria:
        - Choose the states ω_i which are individually most likely. This optimality criterion maximizes the expected number of correct individual states.
        - Choose the state sequence that maximizes the expected number of correct pairs of states ($\omega(t)$, $\omega(t+1)$), or triples of states ($\omega(t)$, $\omega(t+1)$, $\omega(t+1)$), etc.
        - <u>**Most widely used**</u>: find the single best state sequence (path), i.e., maximize $P(\omega^T \mid V^T, \theta)$ which is equivalent to maximizing $P(\omega^T, V^T \mid \theta)$

Maximizing the expected number of correct individual states

The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k.

The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable.



Algorithm 4 (HMM decoding)

```
begin initialize Path = {}, $t = 0$
        for $t \leftarrow t + 1$
            $k = 0, \alpha_0 = 0$
            for $k \leftarrow k + 1$
                $\alpha_k(t) \leftarrow b_{jk} v(t) \sum_{i=1}^{c} \alpha_i(t-1) a_{ij}$
            until $k = c$
            $j' \leftarrow \arg\max_j \alpha_j(t)$
            $AppendTo$ Path $\omega_{j'}$
        until $t = T$
    return Path
end
```

Remarks:

- The problem with choosing the individually most likely states is that the overall state sequence may not be valid
    - Consider a situation where the individually most likely states are $\omega(t) = \omega_i$ and , $\omega(t+1) = \omega_j$, but the transition probability $a_{ij} = 0$
- To avoid this problem, it is common to look for the single best state sequence, at the expense of having sub-optimal individual states
    - This is accomplished with the Viterbi algorithm

Viterbi Algorithm

To find the **single best state sequence** we define yet another variable:

$$\delta_i(t) = \max_{\{\omega(1),\omega(2),\ldots,\omega(t-1)\}} P(\omega(1),\omega(2),\ldots,\omega(t-1),\omega_i(t),v_1,v_2,\ldots,v_t,|\,\theta)$$

which represents the **highest probability along a single path that accounts for the first *t* observations and ends at state ω_i**

By induction, $\delta_j(t+1)$ can be computed as:

$$\delta_j(t+1) = \left[\max_i \delta_i(t) a_{ij}\right] b_{jk}, \ \ where \ \ v_k = v(t)$$

To retrieve the state sequence, we also need to keep track of the state that maximizes $\delta_i(t)$ at each time t, which is done by constructing an array:

$$\Psi_j(t+1) = \arg\max_{1 \le i \le N} \left[\delta_i(t) a_{ij}\right]$$

$\Psi_j(t+1)$ is the state at time t from which a transition to state ω_j maximizes the probability $\delta_j(t+1)$.

Viterbi Algorithm

Initialization (no previous states): $\delta_i(1) = \pi_i b_{ik}, \ \ where \ \ v_k = v(1)$

$$\Psi_i(1) = 0$$

Recursion: For $2 \le t \le T$; $1 \le j \le N$ $\delta_j(t+1) = \left[\max_i \delta_i(t) a_{ij}\right] b_{jk}, \ \ where \ \ v_k = v(t)$

$$\Psi_j(t+1) = \arg\max_{1 \le i \le N} \left[\delta_i(t) a_{ij}\right]$$

Termination: $\omega^*(T) = \arg\max_{1 \le i \le N} \left[\delta_i(T)\right] \ \ ; \ \ P^* = \max_{1 \le i \le N} \left[\delta_i(T)\right]$

And the optimal state sequence can be retrieved by backtracking

$$\omega^*(t) = \Psi_j(t+1), \ \ where \ \omega_j = \omega^*(t+1) \ \ and \ \ t = T-1, T-2, \ldots, 1$$

- **Remark**: the Viterbi algorithm is similar to the Forward procedure, except that it uses a maximization over previous states instead of a summation.

The Learning Problem

This problem is the one in which we attempt to optimize the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence since it is used to “train” the HMM.

The training problem is the crucial one for most applications of HMMs, since it allows us to optimally adapt model parameters to observed training data-i.e., to **create best models for real phenomena.**

How do you design an HMM?

1. Occasionally, it is reasonable to deduce the HMM from first principles.

2. Usually, especially in Speech or Genetics, it is better to infer it from large amounts of data $\{v_1, v_2, ..., v_T\}$ with a big value of T.

Inferring an HMM: Remember, we've been doing things like $P(v_1, v_2, ..., v_T \mid \theta)$ where "θ" is the notation for our HMM parameters.

Options: We could use : 1) MAX LIKELIHOOD: $\theta = \text{argmax } P(v_1, v_2, ..., v_T \mid \theta)$

2) BAYES: Work out $P(\theta \mid v_1, v_2, ..., v_T)$ and then take $E[\theta]$ or $\max P(\theta \mid v_1, v_2, ..., v_T)$

The Forward-Backward Algorithm

The Forward-Backward algorithm is an instance of a generalized Expectation-Maximization algorithm.

Approach: iteratively update the weights in order to better explain the observed training sequences. This is done in two steps

Expectation Step: If we knew θ we could estimate EXPECTATIONS of quantities such as

Expected number of times in state i

Expected number of transitions $i \rightarrow j$

Maximization Step: If we knew the quantities such as :

Expected number of times in state i

Expected number of transitions $i \rightarrow j$

then we could compute the MAX LIKELIHOOD estimate of $\theta = (\{a_{ij}\},\{b_{ij}\}, \pi_i)$

- Previously (the forward algorithm) **we defined** ***$\alpha_i(t)$*** as **the probability that our HMM is in the state ω_i at time t, having generated the first t elements of the observation sequence**.

- Analogously, **we defined** **$\beta_i(t)$** as **the probability that our HMM is in the state ω_i at time t, and will generate the remaining elements of the observation sequence, from $t+1$ to T**.

- Now we define **$\gamma_{ij}(t)$** as **the probability that our HMM is in the state ω_i at time t-1 and in the state ω_j at time t, given the model generated the entire training sequence V^T, by any path.**

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{ij}\beta_i(t)}{P(V^T \mid \theta)}$$

We want the estimate: $\hat{a}_{ij} = P(\omega(t) = \omega_j \mid \omega(t-1) = \omega_i)$

This estimate can be found by taking the ratio between the expected number of transition from ω_i to ω_j and the total expected number of transitions from ω_i .

$$\hat{a}_{ij} = \frac{\text{Expected \# transitions } i \to j \mid \theta^{old}, v_1, v_2, \cdots v_T}{\sum_{k=1}^{N} \text{Expected \# transitions } i \to k \mid \theta^{old}, v_1, v_2, \cdots v_T}$$

$$= \frac{\sum_{t=1}^{T} P(\omega(t) = \omega_j, \omega(t-1) = \omega_i \mid \theta^{\text{old}}, v_1, v_2, \cdots v_T)}{\sum_{k=1}^{N} \sum_{t=1}^{T} P(\omega(t) = \omega_k, \omega(t-1) = \omega_i \mid \theta^{\text{old}}, v_1, v_2, \cdots v_T)}$$

Considering the previous notations, the estimate will be

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_{ik}(t)}$$

In the same way, we can obtain an improved estimate of b_{jk} .Thus we have

$$\hat{b}_{jk} = \frac{\text{Expected \# of times in state j and observing symbol } v_k}{\text{Expected \# of times in state j}}$$

Considering the previous notations , the estimation formula becomes:

$$\hat{b}_{jk} = \frac{\sum\limits_{\substack{t=1,\dots,T \\ v(t)=v_k}} \sum\limits_{i=1}^{N} \gamma_{ji}(t)}{\sum\limits_{t=1}^{T} \sum\limits_{i=1}^{N} \gamma_{ji}(t)}$$

EM for HMMs (BAUM-WELCH algorithm)

1. Get your observations $v_1, v_2, \ldots, v_T$
2. Guess your first θ estimate $\theta(0)$, $k=0$
3. $k = k+1$
4. Given $v_1, v_2, \ldots, v_T$, and $\theta(k-1)$ compute

$$\gamma_{ij}(t) \quad \forall 1 \le t \le T, \quad \forall 1 \le i \le N, \quad \forall 1 \le j \le N$$

5. Compute expected no of transitions out of state i during path, and expected no of transitions from $i \rightarrow j$ during path
6. Compute new estimates of a_{ij}, b_{jk}, π_i accordingly. Call them $\theta(k)$
7. Goto 3, unless converged.

Important

To remember:

1) EM does not estimate the number of states. That must be given. There is a trade-off between too few states (inadequately modeling the structure in the data) and too many (fitting the noise). Thus the number of states is a regularization parameter.)

2) Often, HMMs are forced to have some links with zero probability. This is done by setting $a_{ij}=0$ in initial estimate $\theta(0)$

Applications of HMMs

HMMs can be used as black-box density models on sequences. They have the advantage over Markov models in that they can represent long-range dependencies between observations, mediated via the latent variables.

Applications of HMMs:

- Robot planning + sensing when there’s uncertainty
- Speech Recognition/Understanding
- Human Genome Project
- Consumer decision modeling
- Economics & Finance

References

1. L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. of the IEEE, Vol.77, No.2, pp.257--286, 1989 (http://ieeexplore.ieee.org/iel5/5/698/00018626.pdf?arnumber=18626)