Hidden Markov Models

From Theory to Applications

Alexandru Sorici, Tudor Berariu

Romanian Asociation for Artificial Intelligence

October, 27th, 2012

PART 1.
Intro

PART 2.
Theory of HMMs

PART 3.
Demo & Discussions

Outline

Outline

Outline

Outline

ARIA EDU

:)

Outline

Today's Program

9:00	Registration
10:00	ARIA
11:00	HMM Theory

Outline

Outline

What is Machine Learning?

Machine Learning

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Machine Learning Applications

- Computer Vision: Google Car
- Machine Translation: Google Translate, new Speech-to-Speech technologies
- Speech Recognition: Siri, S Voice
- Recommender Systems: Amazon, Netflix, YouTube
- Intelligent Advertising: every big player :-)

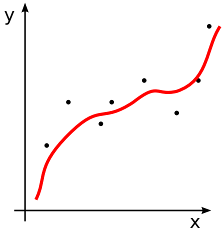
Outline

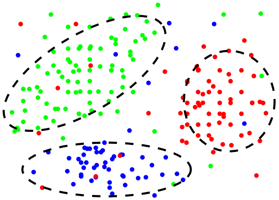
Machine Learning Classification

Types of Machine Learning Problems

- Regression
- Classification
- Reinforcement Learning

- supervised learning (eg. ..)
- unsupervised





Sequence / Temporal problems (I)

OBJECT TRACKING

Speaker Detection

GPS

Robotics

Surface to air missille

Ship or rocket navigation

SPEECH RECOGNITION

Voice User Interfaces - e.g. SIRI

Speech-to-Text Processing

Direct Voice Input - Aircraft

GESTURE RECOGNITION

Personalized Signature Recognition

Human Activity Recognition

Sign Language Recognition

Sequence / Temporal problems (II)

BIOINFORMATICS

Protein Sequencing

Modeling of a Gene
Regulatory Network

ECONOMICS

Stock Price Prediction

Econometrics
- estimate a country's econmic indicators across time -

Probabilistic Reasoning over Time - Models

Consider some of the previously presented problems ...

Probabilistic Reasoning over Time - Models

Consider some of the previously presented problems ...

How do we model such dynamic situations?

Probabilistic Reasoning over Time - Models

Consider some of the previously presented problems ...

How do we model such dynamic situations?

States and Observations

- The process of change is viewed as a series of time slices (snapshots)
- Each time slice contains a set of random variables
    - $\mathbf{O}_t$ - set of all *observable* evidence variables at time t
    - $\mathbf{Q}_t$ - set of all *unobservable / hidden* state variables at time t

Probabilistic Reasoning over Time - Assumptions

Consider some of the previously presented problems ...

Probabilistic Reasoning over Time - Assumptions

Consider some of the previously presented problems ...

What assumptions (if any) do we make?

Probabilistic Reasoning over Time - Assumptions

Consider some of the previously presented problems ...

What assumptions (if any) do we make?

Stationary Process

The process of change is governed by laws that do not themselves change over time.
Implication: we need to specify conditional distributions only for the variables within a *representative* timeslice.

Probabilistic Reasoning over Time - Assumptions

Consider some of the previously presented problems ...

What assumptions (if any) do we make?

Stationary Process

The process of change is governed by laws that do not themselves change over time.
Implication: we need to specify conditional distributions only for the variables within a *representative* timeslice.

Markov Assumption

The current state in a process of change depends only on a finite history of previous states.
Implication: there is a bounded number of “parents” for the variables in each time slice.

$$\mathbf{P}(\mathbf{Q}_t|\mathbf{Q}_{1:t-1}) = \mathbf{P}(\mathbf{Q}_t|\mathbf{Q}_{t-1}) \qquad \mathbf{P}(\mathbf{O}_t|\mathbf{Q}_{1:t}, \mathbf{Q}_{1:t-1}) = \mathbf{P}(\mathbf{O}_t|\mathbf{Q}_t)$$

Probabilistic Reasoning over Time - Inference

What are the basic inference tasks that must be solved?

Probabilistic Reasoning over Time - Inference

What are the basic inference tasks that must be solved?

Filtering (monitoring)

The task of computing the belief state - the posterior distribution over the current state, given all evidence to date.
$\mathbf{P}(\mathbf{Q}_t|\mathbf{o}_{1:t})$

Probabilistic Reasoning over Time - Inference

What are the basic inference tasks that must be solved?

Filtering (monitoring)

The task of computing the belief state - the posterior distribution over the current state, given all evidence to date.
$\mathbf{P}(\mathbf{Q}_t|\mathbf{o}_{1:t})$

Evaluation (likelihood)

The task of computing the likelihood of the evidence up to present.
$\mathbf{P}(\mathbf{o}_{1:t})$

Probabilistic Reasoning over Time - Inference

Prediction

The task of computing the posterior distribution over the future state, given all evidence to date.
$\mathbf{P}(\mathbf{Q}_{t+k} | \mathbf{o}_{1:t})$, for some $k > 0$

Probabilistic Reasoning over Time - Inference

Prediction

The task of computing the posterior distribution over the future state, given all evidence to date.
$\mathbf{P}(\mathbf{Q}_{t+k}|\mathbf{o}_{1:t})$, for some $k > 0$

Smoothing (hindsight)

The task of computing the posterior distribution over a past state, given all evidence to the present.
$\mathbf{P}(\mathbf{Q}_{k}|\mathbf{o}_{1:t})$, for some $1 \leq k < t$
Provides a better estimate of the state than was available at the time.

Probabilistic Reasoning over Time - Inference

Most likely explanation

Given a *sequence of observations*, find the sequence of states that is most likely to have generated those observations. $argmax_{q_{1:t}}\ \mathbf{P}(\mathbf{q}_{t+k}|\mathbf{o}_{1:t})$, for some $k > 0$

Probabilistic Reasoning over Time - Inference

Most likely explanation

Given a *sequence of observations*, find the sequence of states that is most likely to have generated those observations. $argmax_{q_{1:t}}\ \mathbf{P}(\mathbf{q}_{t+k}|\mathbf{o}_{1:t})$, for some $k > 0$

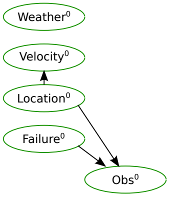
Learning

Given a set of *observation sequences*, find a method to learn the transition (e.g. $\mathbf{P}(\mathbf{q}_{t+1} = s_j|\mathbf{q}_t = s_i)$, $1 \leq i, j < N$) and sensor $(\mathbf{P}(\mathbf{o}_t|\mathbf{q}_t))$ models from the observations.

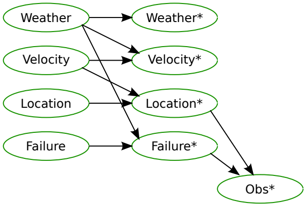
Probabilistic Reasoning over Time - Known Methods

Dynamic Bayesian Networks (DBN)

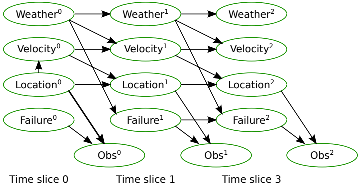
A DBN is Bayesian network that represents a temporal probability model.



(a) the 2-time-slice Bayesian Network



(b) the time 0 network



(c) unrolled DBN over 3 time slices

Figure: A highly simplified DBN for monitoring a vehicle (Koller and Friedman 2009)

Applied in problems like: object tracking, human activity recognition, protein sequencing etc.

Probabilistic Reasoning over Time - Known Methods

Kalman Filters (Linear Dynamical Systems)

A temporal model of one or more real-valued variables that evolve linearly over time, with some Gaussian noise.

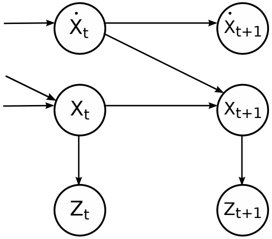


Figure: BN structure for a linear dynamical system with position X_t, velocity $\dot{X}_t$, and position measurement Z_t

- can be viewed as DBNs where all variables are continuous and all dependencies are linear gaussian
- wide application in **object tracking**

Probabilistic Reasoning over Time - Known Methods

Hidden Markov Models (HMM)

An HMM is a temporal probabilistic model in which the state of the process is described by a single discrete random variable. The possible values of the variable are the possible states of the world.

Used successfully in applications like:

- Handwriting Recognition
- Gesture Recognition
- Speech Recognition
- Part-of-Speech Tagging
- DNA Sequencing

Outline

The 3 fundamental problems (Rabiner 1989)

- Particularization of temporal inference problems to the HMM case
- The restricted structure of the HMM allows for elegant implementations of all the basic algorithms

The 3 fundamental problems (Rabiner 1989)

- Particularization of temporal inference problems to the HMM case
- The restricted structure of the HMM allows for elegant implementations of all the basic algorithms

Evaluation Problem

Given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model?

The 3 fundamental problems (Rabiner 1989)

- Particularization of temporal inference problems to the HMM case
- The restricted structure of the HMM allows for elegant implementations of all the basic algorithms

Evaluation Problem

Given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model?

Best Explanation of Observations Problem

Given a model and a sequence of observations how do we choose a corresponding sequence of states which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

The 3 fundamental problems (Rabiner 1989)

- Particularization of temporal inference problems to the HMM case
- The restricted structure of the HMM allows for elegant implementations of all the basic algorithms

Evaluation Problem

Given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model?

Best Explanation of Observations Problem

Given a model and a sequence of observations how do we choose a corresponding sequence of states which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

Model Estimation (Training) Problem

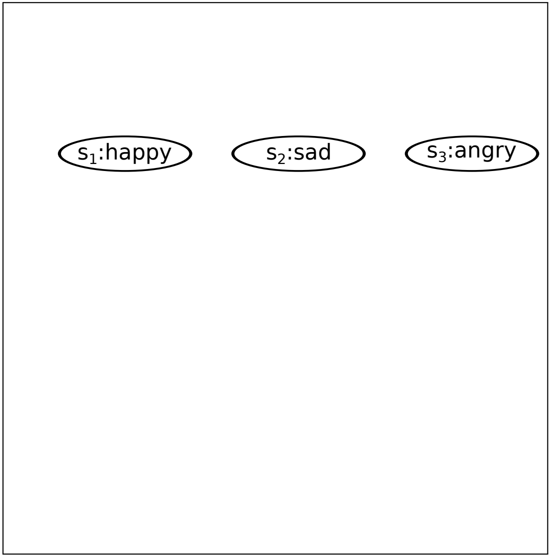
Given some observed sequences, how do we adjust the parameters of an HMM model that best tries to explain the observations?

Outline

An example problem: Emotional states

Let's consider a simple example:
a robot that tracks the emotional states of a player.

An example problem: Emotional states



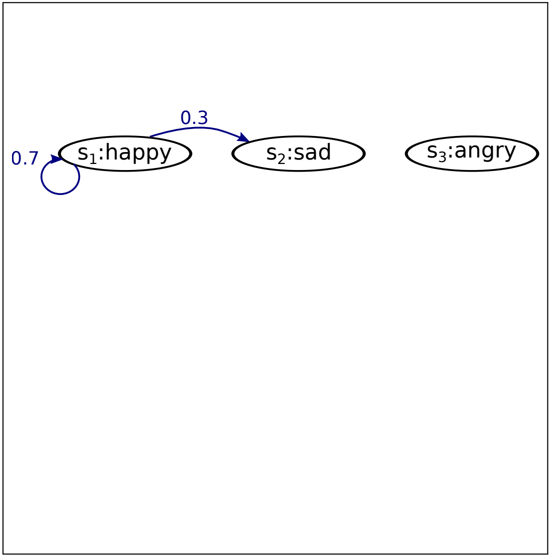
N - number of states

$\mathbf{N} = 3$

states:

- s_1: happy
- s_2: sad
- s_3: angry

An example problem: Emotional states



A - state transition probability distribution

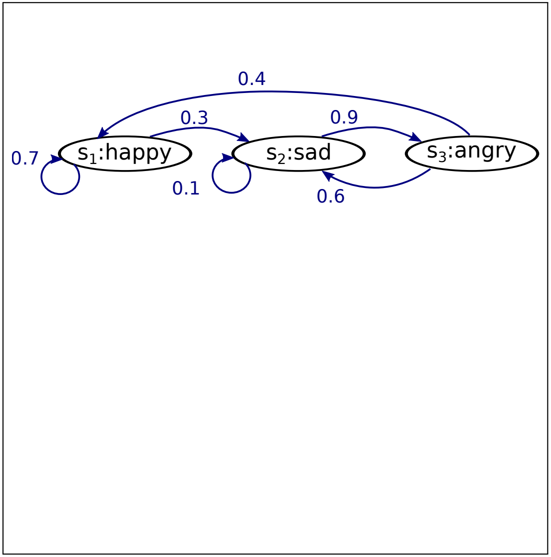
$\mathbf{A} = \{a_{i,j}\},\ 1 \leq i,j \leq N$

$a_{i,j} = P(q_{t+1} = s_j | q_t = s_i)$

- $a_{\mathbf{1},1} = 0.7$
- $a_{\mathbf{1},2} = 0.3$
- $a_{\mathbf{1},3} = 0$

$$\sum_{j=1}^{N} a_{i,j} = 1, \quad 1 \leq i \leq N$$

An example problem: Emotional states



A - state transition probability distribution

$\mathbf{A} = \{a_{i,j}\}, \; 1 \leq i,j \leq N$

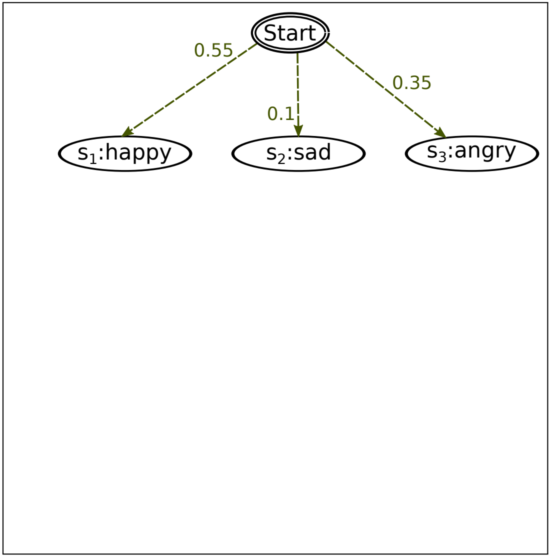
$a_{i,j} = P(q_{t+1} = s_j | q_t = s_i)$

- $a_{\mathbf{1},1} = 0.7$
- $a_{\mathbf{1},2} = 0.3$
- $a_{\mathbf{1},3} = 0$

$$\sum_{j=1}^{N} a_{i,j} = 1, \quad 1 \leq i \leq N$$

$$\mathbf{A} = \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0.7 & 0.3 & 0 \\ s_2 & 0 & 0.9 & 0.1 \\ s_3 & 0.4 & 0.6 & 0 \end{array}$$

An example problem: Emotional states



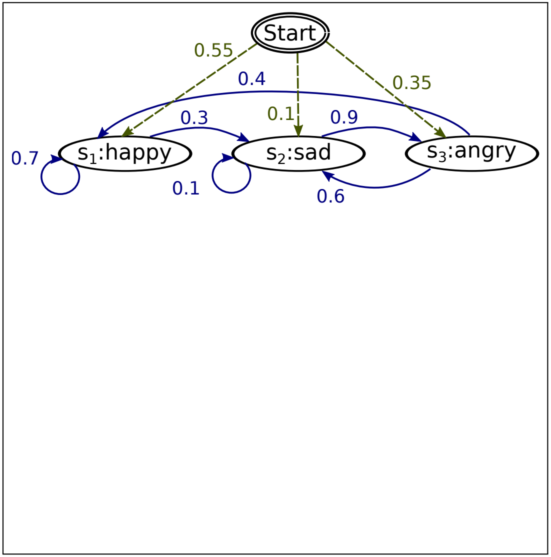
Π - initial state distribution

$\mathbf{\Pi} = \{\pi_i\}, \quad 1 \leq i \leq N$

$\pi_i = P(q_1 = s_i)$

$$\mathbf{\Pi} = \begin{array}{c} s_1 \quad s_2 \quad s_3 \\ \begin{pmatrix} 0.35 & 0.1 & 0.55 \end{pmatrix} \end{array}$$

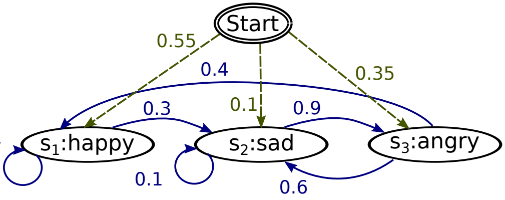
An example problem: Emotional states



$$A = \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0.7 & 0.3 & 0 \\ s_2 & 0 & 0.9 & 0.1 \\ s_3 & 0.4 & 0.6 & 0 \end{array}$$

$$\Pi = \begin{array}{ccc} s_1 & s_2 & s_3 \\ 0.35 & 0.1 & 0.55 \end{array}$$

An example problem: Emotional states

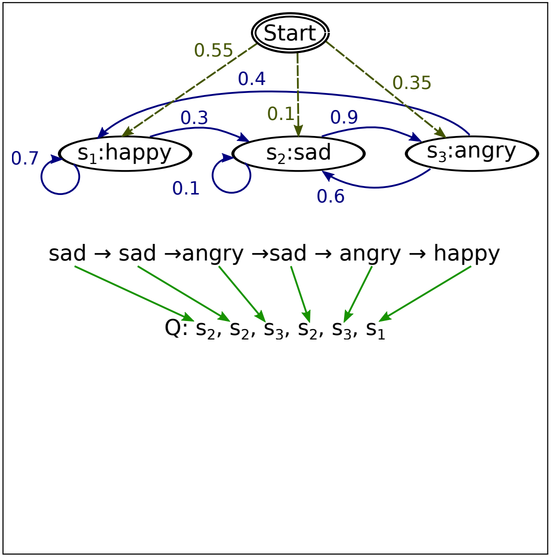


sad → sad →angry →sad → angry → happy

$$A = \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0.7 & 0.3 & 0 \\ s_2 & 0 & 0.9 & 0.1 \\ s_3 & 0.4 & 0.6 & 0 \end{array}$$

$$\Pi = \begin{array}{ccc} s_1 & s_2 & s_3 \\ \hline 0.35 & 0.1 & 0.55 \end{array}$$

An example problem: Emotional states


Start
0.55
0.4
0.35
0.3
0.1
0.9
0.7
s1:happy
s2:sad
s3:angry
0.1
0.6
sad → sad →angry →sad → angry → happy
Q: s2, s2, s3, s2, s3, s1

$$A = \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0.7 & 0.3 & 0 \\ s_2 & 0 & 0.9 & 0.1 \\ s_3 & 0.4 & 0.6 & 0 \end{array}$$

$$\Pi = \begin{array}{ccc} s_1 & s_2 & s_3 \\ 0.35 & 0.1 & 0.55 \end{array}$$

$$Q = [q_1 q_2 \cdots q_T]$$

$$P(Q|A,\Pi) =$$

$$= \pi_{q_1} a_{q_1,q_2} \cdots a_{q_{T-1},q_T}$$

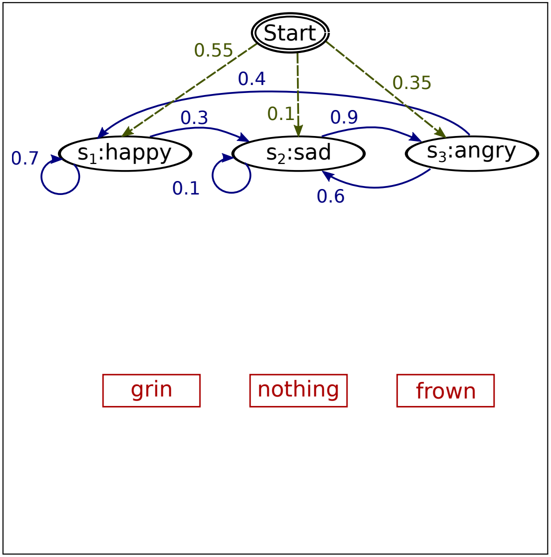
$$P(s_2, s_2, s_3, s_2, s_3, s_1 | A, \Pi) =$$

$$= \pi_2 \cdot a_{2,2} \cdot a_{2,3} \cdot a_{3,2} \cdot a_{2,3} \cdot a_{3,1}$$

$$= 0.1 \cdot 0.3 \cdot 0.1 \cdot 0.9 \cdot 0.6 \cdot 0.9 \cdot 0.4$$

$$= 0.0005832$$

An example problem: Emotional states



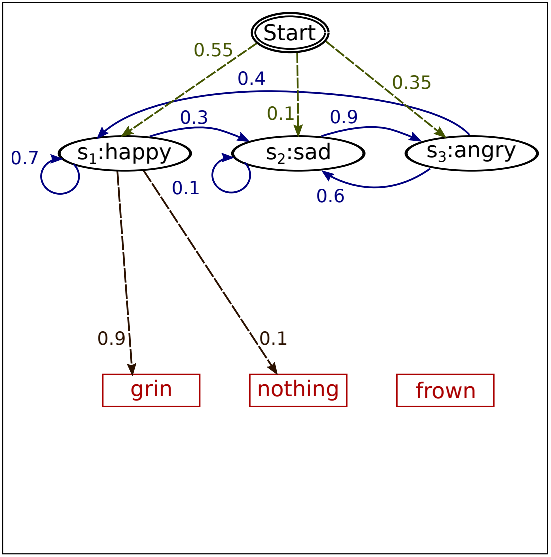
M - number of distinct observable values

$\mathbf{M} = 3$

values:

- v_1: grin
- v_2: nothing
- v_3: frown

An example problem: Emotional states



B - observation values probability distribution

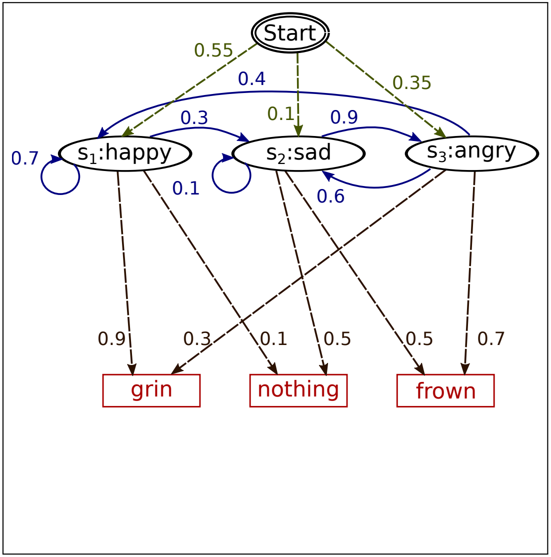
$\mathbf{B} = \{b_{j,k}\}_{1 \le j \le N, 1 \le k, \le M}$

$$b_{j,k} = b_j(v_k) \\ = P(o_t = v_k | q_t = s_j)$$

- $b_{\mathbf{1},1} = b_{\mathbf{1}}(grin) = 0.9$
- $b_{\mathbf{1},2} = b_{\mathbf{1}}(nothing) = 0.1$
- $b_{\mathbf{1},3} = b_{\mathbf{1}}(frown) = 0$

$$\sum_{k=1}^{M} b_{j,k} = 1, \quad 1 \le j \le N$$

An example problem: Emotional states



B - observation values probability distribution

$\mathbf{B} = \{b_{j,k}\}_{1 \le j \le N, 1 \le k, \le M}$

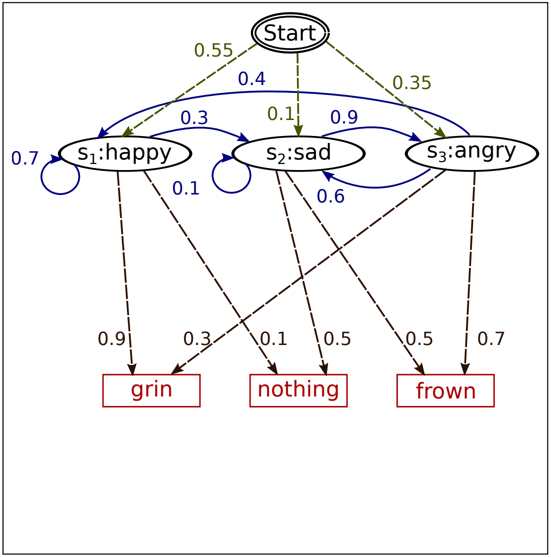
$$b_{j,k} = b_j(v_k)$$
$$= P(o_t = v_k | q_t = s_j)$$

- $b_{\mathbf{1},1} = b_{\mathbf{1}}(grin) = 0.9$
- $b_{\mathbf{1},2} = b_{\mathbf{1}}(nothing) = 0.1$
- $b_{\mathbf{1},3} = b_{\mathbf{1}}(frown) = 0$

$$\sum_{k=1}^{M} b_{j,k} = 1, \quad 1 \le j \le N$$

$$\mathbf{B} = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{ccc} grin & notg & frown \end{array} \\ \begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.5 & 0.5 \\ 0.3 & 0 & 0.7 \end{pmatrix} \end{array}$$

An example problem: Emotional states



λ - parameters of the model

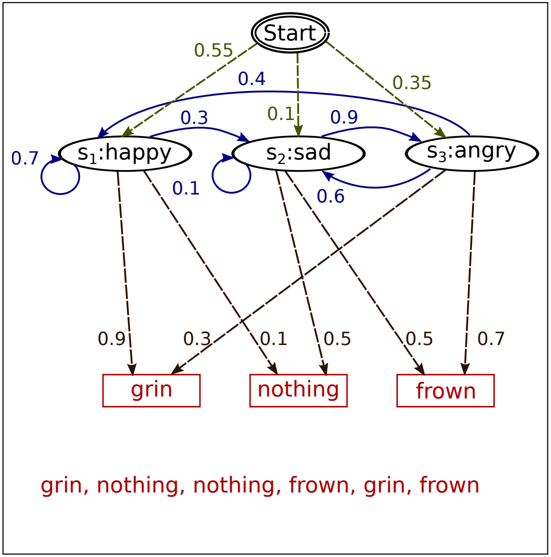
$\lambda = (A, B, \Pi)$

A - state transition probability distribution

B - observation values probability distribution

Π - initial state distribution

An example problem: Emotional states

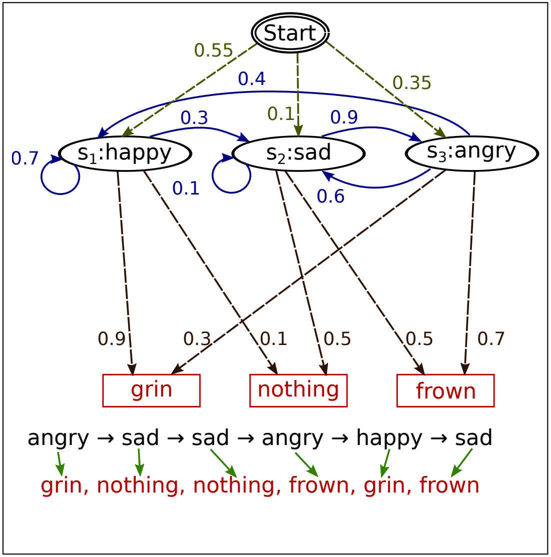


O - observation sequence

T - length of observation sequence

$O = [o_1 o_2 \cdots o_T]$

An example problem: Emotional states

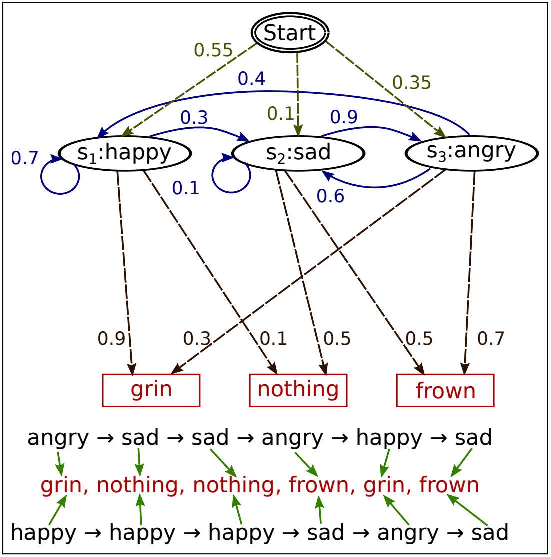


O - observation sequence

T - length of observation sequence

$$O = [o_1 o_2 \cdots o_T]$$

An example problem: Emotional states


Start
0.55
0.1
0.35
0.4
0.3
0.9
0.7
s1:happy
s2:sad
s3:angry
0.1
0.6
0.9
0.3
0.1
0.5
0.5
0.7
grin
nothing
frown
angry → sad → sad → angry → happy → sad
grin, nothing, nothing, frown, grin, frown
happy → happy → happy → sad → angry → sad

O - observation sequence

T - length of observation sequence

$$O = [o_1 o_2 \cdots o_T]$$

An example problem: Emotional states

- Example inspired from:

  R. Zubek (2006). “Introduction to hidden markov models”. In: *AI Game Programming Wisdom* 3, pp. 633–646

Restating the three fundamental HMM Problems

Evaluation Problem

Given a model and a sequence of observations , how do we compute the probability that the observed sequence was produced by the model?

Restating the three fundamental HMM Problems

Evaluation Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations , how do we compute the probability that the observed sequence was produced by the model?

Restating the three fundamental HMM Problems

Evaluation Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$, how do we compute the probability that the observed sequence was produced by the model?

Restating the three fundamental HMM Problems

Evaluation Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$, how do we compute the probability $P(O|\lambda)$ that the observed sequence was produced by the model?

Restating the three fundamental HMM Problems

Evaluation Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$, how do we compute the probability $P(O|\lambda)$ that the observed sequence was produced by the model?

- Enumerate every possible state sequence:

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q, \lambda) \cdot P(Q|\lambda) \qquad (1)$$

Restating the three fundamental HMM Problems

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda) \cdot P(Q|\lambda) \tag{1}$$

Restating the three fundamental HMM Problems

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda) \cdot P(Q|\lambda) \tag{1}$$

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t) = b_{q_1}(o_1) \cdot \ldots \cdot b_{q_T}(o_T) \tag{2}$$

Restating the three fundamental HMM Problems

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda) \cdot P(Q|\lambda) \tag{1}$$

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t) = b_{q_1}(o_1) \cdot \ldots \cdot b_{q_T}(o_T) \tag{2}$$

$$P(Q|\lambda) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1},q_t} = \pi_{q_1} \cdot a_{q_1,q_2} \cdot a_{q_2,q_3} \cdot \ldots \cdot a_{q_{T-1},q_T} \tag{3}$$

Restating the three fundamental HMM Problems

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda) \cdot P(Q|\lambda) \tag{1}$$

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t) = b_{q_1}(o_1) \cdot \ldots \cdot b_{q_T}(o_T) \tag{2}$$

$$P(Q|\lambda) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1},q_t} = \pi_{q_1} \cdot a_{q_1,q_2} \cdot a_{q_2,q_3} \cdot \ldots \cdot a_{q_{T-1},q_T} \tag{3}$$

$$\begin{aligned} P(O|\lambda) &= \sum_{all\ Q} P(O,Q|\lambda) = \sum_{all\ Q} P(O,|Q,\lambda) \cdot P(Q,\lambda) \\ &= \sum_{\text{all } Q} \left(\pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=2}^{T} b_{q_t}(o_t) a_{q_{t-1},q_t} \right) \end{aligned} \tag{1}$$

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model and a sequence of observations how do we choose a corresponding sequence of states which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations how do we choose a corresponding sequence of states which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$ how do we choose a corresponding sequence of states which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$ how do we choose a corresponding sequence of states $Q = [q_1 q_2 \cdots q_T]$ which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$ how do we choose a corresponding sequence of states $Q = [q_1 q_2 \cdots q_T]$ which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

- There is no single answer.
- The sequence of individually most likely states:

$$Q_{\text{best}} = [\hat{q}_1\ \hat{q}_2\ \ldots \hat{q}_T], \quad \hat{q}_t = \underset{s_i}{argmax}\ P(q_t = s_i | O, \lambda) \qquad (4)$$

Restating the three fundamental HMM Problems

Best Explanation of Observations Problem

Given a model $\lambda = (A, B, \Pi)$ and a sequence of observations $O = [o_1 o_2 \cdots o_T]$ how do we choose a corresponding sequence of states $Q = [q_1 q_2 \cdots q_T]$ which *gives meaning* to the observations? How do we *uncover* the hidden part of the model?

- There is no single answer.
- The sequence of individually most likely states:

$$Q_{\text{best}} = [\hat{q}_1\ \hat{q}_2\ \ldots \hat{q}_T], \quad \hat{q}_t = \underset{s_i}{argmax}\ P(q_t = s_i | O, \lambda) \tag{4}$$

- The best path

$$Q_{\text{best}} = \underset{Q}{\text{argmax}}\ P(Q|O, \lambda) = \underset{Q}{\text{argmax}}\ P(Q, O|\lambda) \tag{5}$$

Restating the three fundamental HMM Problems

Model Estimation (Training) Problem

Given some observed sequences , how do we adjust the parameters of an HMM model that best tries to explain the observations?

Restating the three fundamental HMM Problems

Model Estimation (Training) Problem

Given some observed sequences $\mathcal{O} = [O_1 O_2 \cdots O_L]$, how do we adjust the parameters of an HMM model that best tries to explain the observations?

Restating the three fundamental HMM Problems

Model Estimation (Training) Problem

Given some observed sequences $\mathcal{O} = [O_1 O_2 \cdots O_L]$, how do we adjust the parameters $\lambda = (A, B, \Pi)$ of an HMM model that best tries to explain the observations?

Restating the three fundamental HMM Problems

Model Estimation (Training) Problem

Given some observed sequences $\mathcal{O} = [O_1 O_2 \cdots O_L]$, how do we adjust the parameters $\lambda = (A, B, \Pi)$ of an HMM model that best tries to explain the observations?

- The above question can be asked formally:

$$\lambda_{\text{best}} = \underset{\lambda}{\operatorname{argmax}}\, P(\mathcal{O}|\lambda) \tag{6}$$

Outline

Notation Conventions

Variables in Octave

Outline

α (forward) variables

- Can we *efficiently* compute $P(O|\lambda)$?

  Yes, using the **forward-backward** algorithm

α (forward) variables

- Can we *efficiently* compute $P(O|\lambda)$?

  Yes, using the **forward-backward** algorithm

- Introducing α (forward) variables:

$$\underset{1 \leq t \leq T, 1 \leq i \leq N}{\alpha_{t,i} = P(o_1, o_2, \ldots, o_t, q_t = S_i|\lambda)} \tag{7}$$

α (forward) variables

- Can we *efficiently* compute $P(O|\lambda)$?

  Yes, using the **forward-backward** algorithm

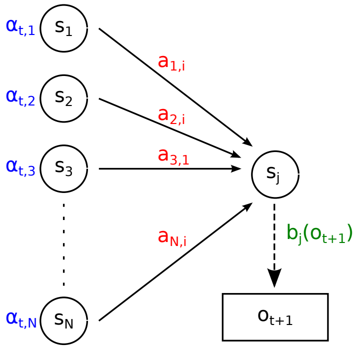
- Introducing α (forward) variables:

$$\alpha_{t,i} = P(o_1, o_2, \ldots, o_t, q_t = S_i|\lambda) \\ {}_{1 \le t \le T, 1 \le i \le N} \tag{7}$$

- Relation between $P(O|\lambda)$ and α variables:

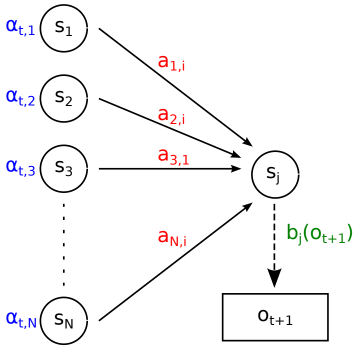
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_{T,i} \tag{8}$$

Computing α variables



- α variables initialization
  $P(o_1, q_1 = s_i) = P(o_1 | q_1 = s_i) P(q_1 = s_i)$
  $\alpha_{1,i} = \pi_i b_i(o_1), \quad 1 \leq i \leq N$

Computing α variables



- α variables initialization
  $P(o_1, q_1 = s_i) = P(o_1|q_1 = s_i)P(q_1 = s_i)$
  $\alpha_{1,i} = \pi_i b_i(o_1), \quad 1 \leq i \leq N$
- Induction step

$$\alpha_{t+1,j} = \left[\sum_{i=1}^{N} \alpha_{t,j} a_{i,j}\right] b_j(o_{t+1}), \qquad \substack{1 \leq t \leq T-1, \\ 1 \leq j \leq N}$$

- Probability of the observed sequence

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_{T,i}$$

β (backward) variables

- Introducing β (backward) variables:

$$\beta_{t,i} = P(o_{t+1}o_{t+2}\cdots o_T | q_t = S_i, \lambda) \tag{9}$$

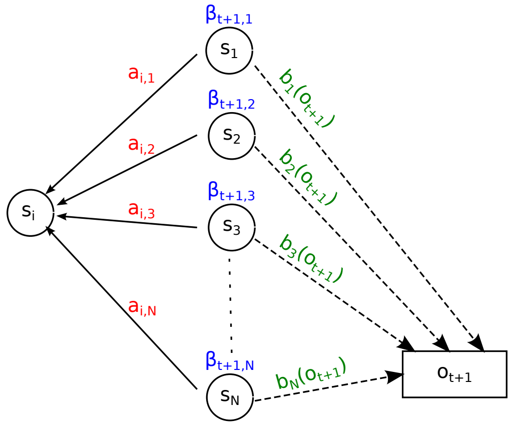
β (backward) variables

- Introducing β (backward) variables:

$$\beta_{t,i} = P(o_{t+1} o_{t+2} \cdots o_T | q_t = S_i, \lambda) \tag{9}$$

- β variables are not needed to compute $P(O|\lambda)$, but they are useful for the other two problems
- β variables can be computed in a similar (efficient) way to the procedure for the α variables

Computing β variables



- β variables initialization
  $\beta_{T,i} = 1, \quad 1 \le i \le N$

- Induction step

$$\beta_{t,i} = \sum_{j=1}^{N} a_{i,j} b_j(o_{t+1}) \beta_{t+1,j}, \quad t = T-1, T-2, \ldots, 1, 1 \le i \le N$$

Scaling problems

- Remember $P(O|\lambda)$:

$$P(O|\lambda) = \sum_{\text{all } Q} \left(\pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=2}^{T} b_{q_t}(o_t) a_{q_{t-1},q_t} \right)$$

Scaling problems

- Remember $P(O|\lambda)$:

$$P(O|\lambda) = \sum_{\text{all } Q} \left(\pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=2}^{T} b_{q_t}(o_t) a_{q_{t-1},q_t} \right)$$

- for large sequences, terms are very close to zero and exceed precision range
- a scaling mechanism is needed

The Forward-Backward algorithm with scaling

- $\hat{\alpha}_{t,i}$ - scaled α variables
- $\hat{\beta}_{t,i}$ - scaled β variables
- C_t - scaling coefficients
- Scaled α variables

$$\bar{\alpha}_{t,i} = C_t \cdot \alpha_{t,i} \tag{10}$$

- Scaled β variables

$$\bar{\beta}_{t,j} = C_t \cdot \beta_{t,j} \tag{11}$$

Computing scaled values

- *Scaled* intialization:

$$[r]\ddot{\alpha}_{1,i} = \alpha_{1,i}, \quad 1 \leq i \leq N \tag{12}$$

$$c_1 = \frac{1}{\sum_{i=1}^{N} \ddot{\alpha}_{1,i}} \tag{13}$$

$$\hat{\alpha}_{1,i} = c_1 \cdot \ddot{\alpha}_{1,i}, \quad 1 \leq i \leq N \tag{14}$$

Computing scaled values

- *Scaled* intialization:

$$[r]\ddot{\alpha}_{1,i} = \alpha_{1,i}, \quad 1 \leq i \leq N \tag{12}$$

$$c_1 = \frac{1}{\sum_{i=1}^{N} \ddot{\alpha}_{1,i}} \tag{13}$$

$$\hat{\alpha}_{1,i} = c_1 \cdot \ddot{\alpha}_{1,i}, \quad 1 \leq i \leq N \tag{14}$$

- *Scaled* induction step:

$$[r]\ddot{\alpha}_{t+1,i} = \left[\sum_{i=1}^{N} \hat{\alpha}_{t,i} a_{i,j}\right] b_j(o_{t+1}) \tag{15}$$

$$c_{t+1} = \frac{1}{\sum_{i=1}^{N} \ddot{\alpha}_{t+1,i}} \tag{16}$$

$$\hat{\alpha}_{t+1,i} = c_{t+1} \cdot \ddot{\alpha}_{t+1,i}, \quad 1 \leq i \leq N \tag{17}$$

Computing $P(O|\lambda)$

- Introducing scale factors prevents exceeding the double precision
- The $P(O|\lambda)$ is related to the scaling factors:

$$P(O|\lambda) = \frac{1}{C_T} = \prod_{t=1} Tc_t \tag{18}$$

The forward-backward algorithm

Algorithm 1 Compute α variables

```
    for $i = 1$ to $N$ do
2:      $\ddot{\alpha}_{1,i} \leftarrow \pi_i \cdot b_i(o_1)$
    end for
4:  $c_1 \leftarrow (\sum_{i=1}^{N} \ddot{\alpha}_{1,i})^{-1}$
    for $i = 1$ to $N$ do
6:      $\hat{\alpha}_{1,i} \leftarrow c_1 \cdot \ddot{\alpha}_{1,i}$
    end for
8:  for $t = 1$ to $T - 1$ do
        for $i = 1$ to $N$ do
10:         $\ddot{\alpha}_{t+1,i} \leftarrow \left[\sum_{i=1}^{N} \hat{\alpha}_{t,i} a_{i,j}\right] b_j(o_{t+1})$
        end for
12:     $c_{t+1} \leftarrow (\sum_{i=1}^{N} \ddot{\alpha}_{t+1,i})^{-1}$
        for $i = 1$ to $N$ do
14:         $\hat{\alpha}_{t+1,i} \leftarrow c_{t+1} \cdot \ddot{\alpha}_{t+1,i}$
        end for
16: end for
```

Algorithm 2 Compute $P(O|\lambda)$

```
    $P \leftarrow \prod_{t=1}^{T} c_t$
```

Algorithm 3 Compute β variables

```
    for $i = 1$ to $N$ do
2:      $\hat{\beta}_{T,i} \leftarrow \cdot c_T$
    end for
4:  for $t = (T - 1)$ to $1$ do
        for $i = 1$ to $N$ do
6:          $\hat{\beta}_{t,i} \leftarrow \sum_{j=1}^{N} a_{i,j} b_j(o_{t+1}) \hat{\beta}_{t+1,j} \cdot c_t$
        end for
8:  end for
```

Let's write some code

- You will implement now the forward-backward algorithm in Octave

Outline

Learning from observations - Reminder

Model Estimation (Training) Problem

Given some observed sequences, how do we adjust the parameters of an HMM model that best tries to explain the observations?

Learning from observations - Reminder

Model Estimation (Training) Problem

Given some observed sequences, how do we adjust the parameters of an HMM model that best tries to explain the observations?

Adjust the model parameters $\lambda = (A, B, \Pi)$ to obtain $\max_{\lambda} P(O|\lambda)$

Learning from observations - Reminder

Model Estimation (Training) Problem

Given some observed sequences, how do we adjust the parameters of an HMM model that best tries to explain the observations?

Adjust the model parameters $\lambda = (A, B, \Pi)$ to obtain $\max_\lambda P(O|\lambda)$

The observation sequence used to adjust the model parameters is called a training sequence.
Training problem is crucial - allows to create best models for real phenomena.

Learning from observations - Aspects of the approach

Learning from observations - Aspects of the approach

Problem

There is no known way to analytically solve for the model which maximizes the probability of the observation sequence.

Learning from observations - Aspects of the approach

Problem

There is no known way to analytically solve for the model which maximizes the probability of the observation sequence.

Solution

We can choose $\lambda = (A, B, \Pi)$ such that $\max_\lambda P(O|\lambda)$ is locally maximized using an iterative procedure such as *Baum-Welch*.
The method is an instance of the *EM algoritm* (Dempster, Laird, and Rubin 1977) for HMMs.

Baum-Welch algorithm (I)

Baum-Welch algorithm (I)

We first define some auxiliary variables:

$\xi_{t,i,j} = \xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda)$
The probability of being in state s_i at time t and in state s_j at time $t+1$, given the model and the observation sequence.

Baum-Welch algorithm (I)

We first define some auxiliary variables:

$\xi_{t,i,j} = \xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda)$
The probability of being in state s_i at time t and in state s_j at time $t+1$, given the model and the observation sequence.

$\gamma_{t,i} = \gamma_t(i) = P(q_t = s_i | O, \lambda)$
The probability of being in state s_i at time t, given the model and the observation sequence.

Baum-Welch algorithm (I)

We first define some auxiliary variables:

$\xi_{t,i,j} = \xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda)$
The probability of being in state s_i at time t and in state s_j at time $t+1$, given the model and the observation sequence.

$\gamma_{t,i} = \gamma_t(i) = P(q_t = s_i | O, \lambda)$
The probability of being in state s_i at time t, given the model and the observation sequence.

From the definitions it follows that:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

Baum-Welch algorithm (II)

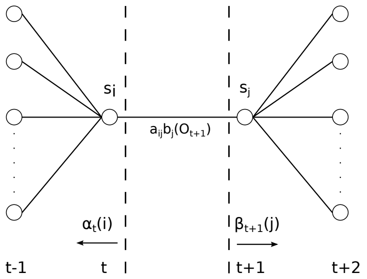


Figure: Sequence of operations required for the computation of the joint event that the system is in state S_i at time t and state S_j at time $t+1$ (Rabiner 1989)

$$\alpha_{t,i} = P(o_1, o_2, \ldots, o_t, q_t = S_i|\lambda)$$

$$\beta_{t,i} = P(o_{t+1}o_{t+2}\cdots o_T|q_t = S_i, \lambda)$$

$$\xi_t(i,j) = \frac{\alpha_{t,i}\cdot a_{i,j}\cdot b_j(o_{t+1})\cdot\beta_{t+1,j}}{P(O|\lambda)}$$

$$= \frac{\alpha_{t,i}\cdot a_{i,j}\cdot b_j(o_{t+1})\cdot\beta_{t+1,j}}{\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_{t,k}\cdot a_{k,l}\cdot b_l(o_{t+1})\cdot\beta_{t+1,l}}$$

Baum-Welch algorithm (III)

How do these auxiliary variables help?

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } S_i \text{ to } S_j$$

Baum-Welch algorithm (IV)

$$\bar{\pi}_i = \text{ expected no. of times in state } S_i \text{ at time } (t = 1) = \gamma_t(i)$$

Baum-Welch algorithm (IV)

$$\bar{\pi}_i = \text{ expected no. of times in state } S_i \text{ at time } (t = 1) = \gamma_t(i)$$

$$\bar{a_{i,j}} = \frac{\text{expected no. of transitions from } S_i \text{ to } S_j}{\text{expcted no. of transition from } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Baum-Welch algorithm (IV)

$$\bar{\pi}_i = \text{ expected no. of times in state } S_i \text{ at time } (t = 1) = \gamma_t(i)$$

$$\bar{a_{i,j}} = \frac{\text{expected no. of transitions from } S_i \text{ to } S_j}{\text{expcted no. of transition from } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b_{j,k}} = \frac{\text{expected no. of times in } S_j \text{ observing symbol } v_k}{\text{expcted no. of times in } S_j} = \frac{\sum_{t=1, O_t = v_k}^{T} \gamma_t(j)}{\sum^{T}}$$

Baum-Welch algorithm (V)

The routine for the general case:

```
Initialize uniform $\pi_i$ for $1 \leq i \leq N$
Initialize random (stochastic) $a_{i,j}$
Initialize uniform $b_{j,k}$ for $1 \leq k \leq M$

Repeat until convergence
    E step:
        compute auxiliary variables $\xi_t(i,j)$ and $\gamma_t(i)$
        using current $\pi_i$, $a_{i,j}$ and $b_{j,k}$

    M step:
        compute updated parameter models $\bar{\pi}_i$, $\bar{a}_{i,j}$, $\bar{b}_{j,k}$
```

Baum-Welch Iterative Update

Baum-Welch - Let's write some code

LET'S WRITE SOME CODE :-)

Outline

Solve the best explanation problem

- How can we answer the *best explanation* problem?

Solve the best explanation problem

- How can we answer the *best explanation* problem?
- **Individually** most likely states

$$\gamma_{t,i} = P(q_t = s_i | O, \lambda) \tag{19}$$

Solve the best explanation problem

- How can we answer the *best explanation* problem?
- **Individually** most likely states

$$\gamma_{t,i} = P(q_t = s_i | O, \lambda) \tag{19}$$

- Computation

$$\gamma_{t,i} = \frac{\alpha_{t,i}\beta_{t,i}}{P(O|\lambda)} = \frac{\alpha_{t,i}\beta_{t,i}}{\sum_{k=1}^{N} \alpha_{t,k}\beta_{t,k}} \tag{20}$$

Solve the best explanation problem

- How can we answer the *best explanation* problem?
- **Individually** most likely states

$$\gamma_{t,i} = P(q_t = s_i | O, \lambda) \tag{19}$$

- Computation

$$\gamma_{t,i} = \frac{\alpha_{t,i}\beta_{t,i}}{P(O|\lambda)} = \frac{\alpha_{t,i}\beta_{t,i}}{\sum_{k=1}^{N} \alpha_{t,k}\beta_{t,k}} \tag{20}$$

- Problems?

Better optimality criterion

- Can we find a better optimality criterion?

Better optimality criterion

- Can we find a better optimality criterion?
- Single best path
  $Q_{\text{best}} = [\hat{q}_1 \hat{q}_2 \cdots \hat{q}_T]$

$$Q_{\text{best}} = \underset{Q}{\operatorname{argmax}}\, P(Q|O,\lambda) = \underset{Q}{\operatorname{argmax}}\, P(Q,O|\lambda) \tag{6}$$

- **Viterbi algorithm** - dynamic programming

δ variables

- Introducing δ variables:

$$\delta_{t,i} = \max_{q_1,\ldots,q_{t-1}} P([q_1 q_2 \ldots q_{t-1} s_i], [o_1, o_2, \ldots o_t] | \lambda) \tag{21}$$

- $\delta_{t,i}$ - the highest probability for a sequence of t states that ends in s_i which accounts for the first t observations

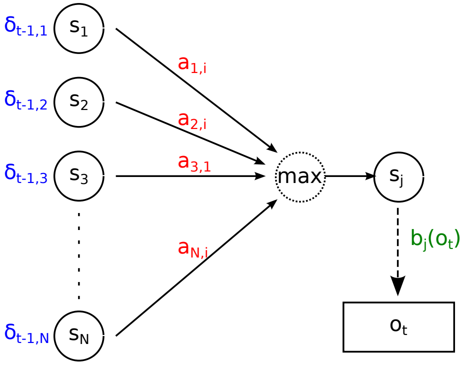
δ variables

- Introducing δ variables:

$$\delta_{t,i} = \max_{q_1,\dots,q_{t-1}} P([q_1 q_2 \dots q_{t-1} s_i], [o_1, o_2, \dots o_t] | \lambda) \tag{21}$$

- $\delta_{t,i}$ - the highest probability for a sequence of t states that ends in s_i which accounts for the first t observations
- the relation between *sequential* δ variables:

$$\delta_{t,j} = [\max_i \delta_{t-1,i} \cdot a_{i,j}] \cdot b_j(o_t) \tag{22}$$



Viterbi algorithm (I)

1 Initialization:

$$\begin{aligned} \delta_{1,i} &= \pi_i b_i(o_1), \quad 1 \leq i \leq N \\ \psi_{1,i} &= 0 \end{aligned} \tag{23}$$

2 Recursion:

$$\begin{aligned} \delta_{t,j} &= [\max_i \delta_{t-1,i} \cdot a_{i,j}] \cdot b_j(o_t) \quad {\scriptstyle 2 \leq t \leq T, 1 \leq j \leq N} \\ \psi_{t,i} &= \underset{i}{argmax}\ \delta t - 1, i \cdot a_{i,j} \quad {\scriptstyle 2 \leq t \leq T, 1 \leq j \leq N} \end{aligned} \tag{24}$$

Viterbi algorithm (II)

3 Termination:

$$
\begin{aligned}
P(Q_{\text{best}} | O, \lambda) &= \max_i \delta_{T,i} \\
\hat{q}_T &= \operatorname*{argmax}_i \delta_{T,i}
\end{aligned}
\tag{25}
$$

4 Backtracking:

$$\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1}), \quad {\scriptstyle t=T-1,T-2,\cdots,1} \tag{26}$$

Outline

A simple symbol recognition application

Features:

A simple symbol recognition application

Features:

Define

Define, organize and visualize a dataset of symbols defined with mouse movements.

A simple symbol recognition application

Features:

Define

Define, organize and visualize a dataset of symbols defined with mouse movements.

Train

Train a HMM-based recognition engine on a symbol dataset.

A simple symbol recognition application

Features:

Define

Define, organize and visualize a dataset of symbols defined with mouse movements.

Train

Train a HMM-based recognition engine on a symbol dataset.

Recognize

Recognize new symbols and view classification metrics.

A simple symbol recognition application

Features:

Define

Define, organize and visualize a dataset of symbols defined with mouse movements.

Train

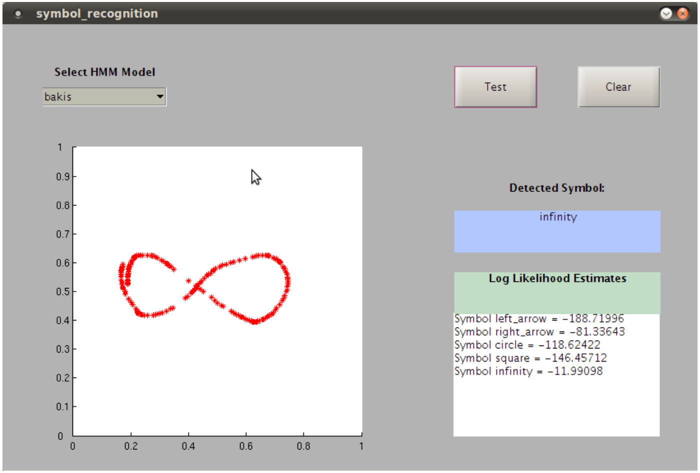
Train a HMM-based recognition engine on a symbol dataset.

Recognize

Recognize new symbols and view classification metrics.

Default included symbols: **left arrow, right arrow, circle, square, infinity**

A simple symbol recognition application - A View

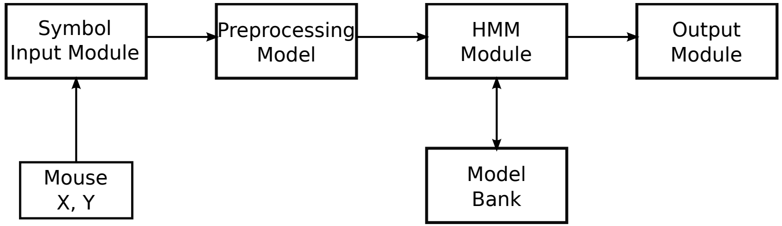


```
Symbol left_arrow = -188.71996
Symbol right_arrow = -81.33643
Symbol circle = -118.62422
Symbol square = -146.45712
Symbol infinity = -11.99098
```

Figure: A view of the symbol recognition application GUI

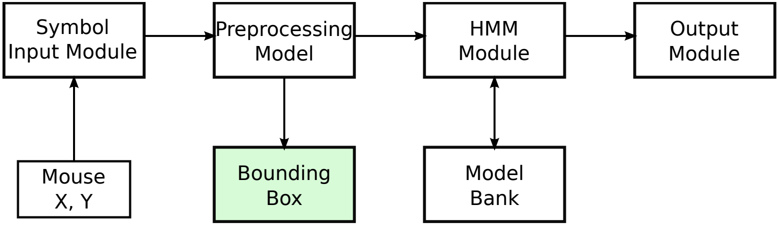
A simple symbol recognition application - Approach (I)

Adapted from (Yang and Xu 1994).



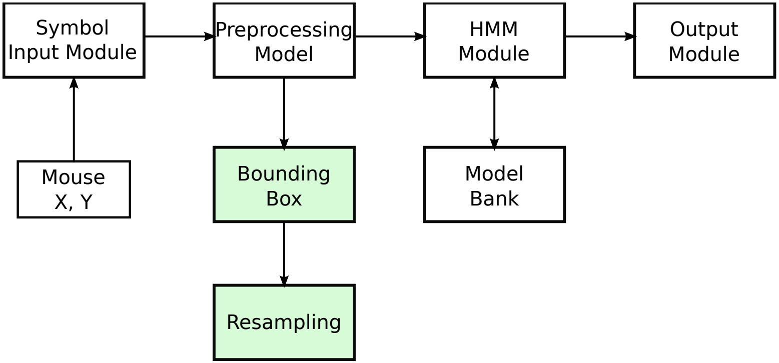
A simple symbol recognition application - Approach (I)

Adapted from (Yang and Xu 1994).



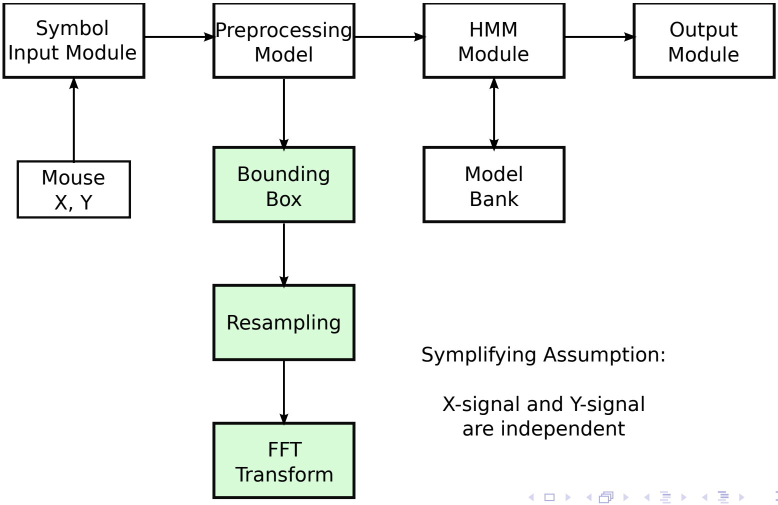
A simple symbol recognition application - Approach (I)

Adapted from (Yang and Xu 1994).



A simple symbol recognition application - Approach (I)

Adapted from (Yang and Xu 1994).



Symplifying Assumption:

X-signal and Y-signal are independent

A simple symbol recognition application - Approach (II)

Adapted from (Yang and Xu 1994).

HMM Structure

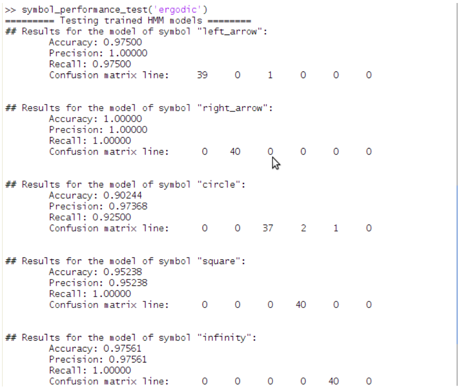
N(number of states) $= 8$
2 discrete observable variables per state - $coef_{FFT}(x)$, $coef_{FFT}(y)$
M(number of values for each observable variable) $= 256$
Transition model:

- Bakis
- Ergodic

A simple symbol recognition application - Results

Dataset size

5 symbols: **left arrow, right arrow, circle, square, infinity**
100 samples per symbol: **50** training, **10** validation, **40** testing



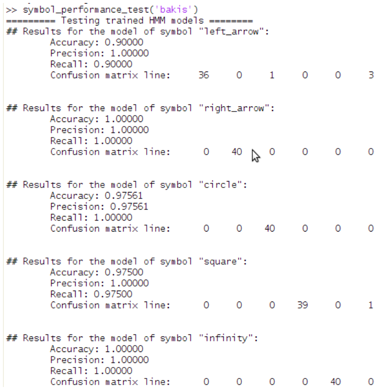
```
>> symbol_performance_test('ergodic')
========= Testing trained HMM models ========
## Results for the model of symbol "left_arrow":
        Accuracy: 0.97500
        Precision: 1.00000
        Recall: 0.97500
        Confusion matrix line:     39     0     1     0     0     0


## Results for the model of symbol "right_arrow":
        Accuracy: 1.00000
        Precision: 1.00000
        Recall: 1.00000
        Confusion matrix line:      0    40     0     0     0     0


## Results for the model of symbol "circle":
        Accuracy: 0.90244
        Precision: 0.97368
        Recall: 0.92500
        Confusion matrix line:      0     0    37     2     1     0


## Results for the model of symbol "square":
        Accuracy: 0.95238
        Precision: 0.95238
        Recall: 1.00000
        Confusion matrix line:      0     0     0    40     0     0


## Results for the model of symbol "infinity":
        Accuracy: 0.97561
        Precision: 0.97561
        Recall: 1.00000
        Confusion matrix line:      0     0     0     0    40     0
```



```
>> symbol_performance_test('bakis')
========= Testing trained HMM models ========
## Results for the model of symbol "left_arrow":
        Accuracy: 0.90000
        Precision: 1.00000
        Recall: 0.90000
        Confusion matrix line:     36     0     1     0     0     3


## Results for the model of symbol "right_arrow":
        Accuracy: 1.00000
        Precision: 1.00000
        Recall: 1.00000
        Confusion matrix line:      0    40     0     0     0     0


## Results for the model of symbol "circle":
        Accuracy: 0.97561
        Precision: 0.97561
        Recall: 1.00000
        Confusion matrix line:      0     0    40     0     0     0


## Results for the model of symbol "square":
        Accuracy: 0.97500
        Precision: 1.00000
        Recall: 0.97500
        Confusion matrix line:      0     0     0    39     0     1


## Results for the model of symbol "infinity":
        Accuracy: 1.00000
        Precision: 1.00000
        Recall: 1.00000
        Confusion matrix line:      0     0     0     0    40     0
```