Training GMMs and HMMs

Multimodal Interaction Lab

b ɜː m ɪ ŋ ə m

UNIVERSITY OF BIRMINGHAM

Outline

- Parameter estimation
- Maximum Likelihood (ML) parameter estimation
- ML for Gaussian PDFs
- ML for GMMs
- ML for HMMs – the Baum-Welch algorithm
- HMM adaptation:
  - MAP estimation
  - MLLR

Multimodal Interaction Lab

b ɜː m ɪ ŋ ə m

UNIVERSITY OF BIRMINGHAM

Discrete variables

- Suppose that Y is a *random variable* which can take any value in a discrete set $X=\{x_1,x_2,\ldots,x_M\}$
- Suppose that $y_1,y_2,\ldots,y_N$ are samples of the random variable Y
- If c_m is the number of times that the $y_n = x_m$ then an estimate of the probability that y_n takes the value x_m is given by:

$$P(x_m) = P(y_n = x_m) \approx \frac{c_m}{N}$$

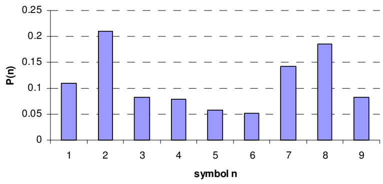
Multimodal Interaction Lab

b ɜː m ɪ ŋ ə m

UNIVERSITY OF BIRMINGHAM

Discrete Probability Mass Function

Symbol	Num.Occurrences
1	120
2	231
3	90
4	87
5	63
6	57
7	156
8	203
9	91
Total	1098



Continuous Random Variables

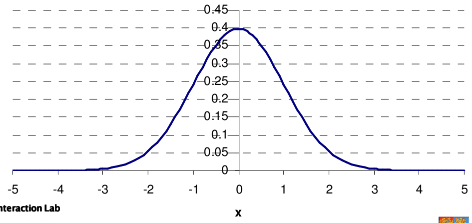
- In most practical applications the data are not restricted to a finite set of values – they can take any value in *N*-dimensional space
- Simply counting the number of occurrences of each value is no longer a viable way of estimating probabilities…
- …but there are generalisations of this approach which are applicable to continuous variables – these are referred to as <u>non-parametric methods</u>

Continuous Random Variables

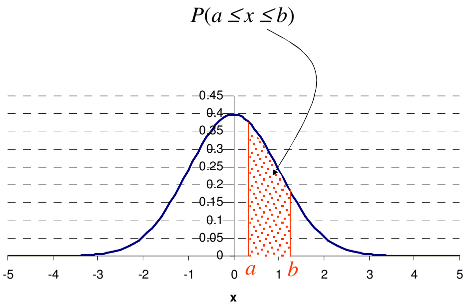
- An alternative is to use a <u>parametric</u> model
- In a parametric model, probabilities are defined by a small set of parameters
- Simplest example is a <u>normal</u>, or <u>Gaussian</u> model
- A Gaussian probability density function (PDF) is defined by two parameters
  - its <u>mean</u> μ, and
  - <u>variance</u> σ

Gaussian PDF

- ‘Standard’ 1-dimensional Guassian PDF:
  - mean μ=0
  - variance σ=1



Gaussian PDF



Gaussian PDF

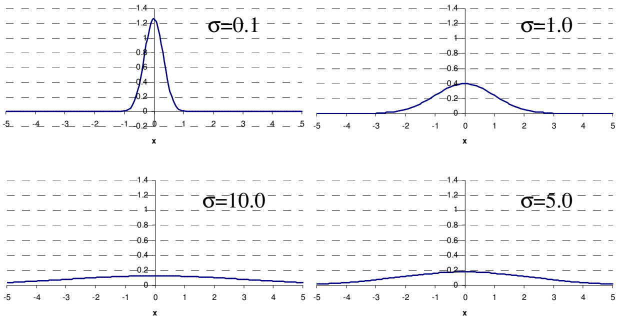
- For a 1-dimensional Gaussian PDF *p* with mean μ and variance σ:

$$p(x) = p(x \mid \mu, \sigma) = \boxed{\frac{1}{\sqrt{2\pi\sigma}}}\boxed{\exp\left(-\frac{(x-\mu)^2}{2\sigma}\right)}$$

Constant to ensure area under curve is 1

Defines ‘bell’ shape

More examples



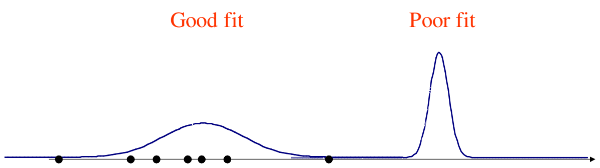
Fitting a Gaussian PDF to Data

- Suppose $y = y_1, \ldots, y_n, \ldots, y_T$ is a sequence of T data values
- Given a Gaussian PDF p with mean μ and variance σ, define:

$$p(y \mid \mu, \sigma) = \prod_{t=1}^{T} p(y_t \mid \mu, \sigma)$$

- How do we choose μ and σ to maximise this probability?

Fitting a Gaussian PDF to Data



Maximum Likelihood Estimation

- Define the best fitting Gaussian to be the one such that $p(y|\mu,\sigma)$ is maximised.
- Terminology:
  - $p(y|\mu,\sigma)$ as a function of y is the probability (density) of y
  - $p(y|\mu,\sigma)$ as a function of μ,σ is the likelihood of μ,σ
- Maximising $p(y|\mu,\sigma)$ with respect to μ,σ is called Maximum Likelihood (ML) estimation of μ,σ

ML estimation of μ,σ

- Intuitively:
  - The maximum likelihood estimate of μ should be the average value of $y_1,\ldots,y_T$, (the sample mean)
  - The maximum likelihood estimate of σ should be the variance of $y_1,\ldots,y_T$, (the sample variance)
- This turns out to be true: $p(y|\ \mu,\ \sigma)$ is maximised by setting:

$$\mu = \frac{1}{T}\sum_{t=1}^{T} y_t, \quad \sigma = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \mu\right)^2$$

Proof

First note that maximising $p(y)$ is the same as maximising $\log(p(y))$

$$\log p(y \mid \mu,\sigma) = \log \prod_{t=1}^{T} p(y_t \mid \mu,\sigma) = \sum_{t=1}^{T} \log p(y_t \mid \mu,\sigma)$$

Also

$$\log p(y_t \mid \mu,\sigma) = -\frac{1}{2}\log(2\pi\sigma) - \frac{(\mu - y_t)^2}{\sigma}$$

At a maximum:

$$0 = \frac{\partial}{\partial\mu}\log p(y \mid \mu,\sigma) = \sum_{t=1}^{T}\frac{\partial}{\partial\mu}\log p(y_t \mid \mu,\sigma) = \sum_{t=1}^{T}\frac{-2(\mu - y_t)(-1)}{\sigma}$$

So, $T\mu = \sum_{t=1}^{T} y_t,\ \mu = \frac{1}{T}\sum_{t=1}^{T} y_t$

ML training for GMMs

- Now consider
  - A Gaussian Mixture Model with M components has
    - M means: $\mu_1,\ldots,\mu_M$
    - M variances $\sigma_1,\ldots,\sigma_M$
    - M mixture weights $w_1,\ldots,w_M$
  - A training sequence $y_1,\ldots,y_T$
- How do we find the maximum likelihood estimate of $\mu_1,\ldots,\mu_M,\ \sigma_1,\ldots,\sigma_M,\ w_1,\ldots,w_M$?

GMM Parameter Estimation

- If we knew which component each sample y_t came from, then parameter estimation would be easy
  - Set μ_m to be the average value of the samples which belong to the m^{th} component
  - Set σ_m to be the variance of the samples which belong to the m^{th} component
  - Set w_m to be the proportion of samples which belong to the m^{th} component
- But we don’t know which component each sample belongs to

Solution – the E-M Algorithm (1)

- Guess initial values

$$\mu_1^{(0)},\ldots,\mu_M^{(0)},\sigma_1^{(0)},\ldots,\sigma_M^{(0)},w_1^{(0)},\ldots,w_M^{(0)}$$

1. For each m calculate the probabilities

$$p_m(y_t) = p\left(y_t \mid \mu_m^{(0)},\sigma_m^{(0)}\right)$$

2. Use these probabilities to estimate how much each sample y_t ‘belongs to’ the m^{th} component

$$\lambda_{m,t} = P(m \mid y_t)$$

Solution – the E-M Algorithm (2)

3. Calculate the new GMM parameters

$$\mu_m^{(1)} = \frac{\sum_{t=1}^{T} \boxed{\lambda_{m,t}}\, y_t}{\sum_{t=1}^{T} \lambda_{m,t}}$$

This is a measure of how much y_t 'belongs to' the m^{th} component

$$\sigma_m^{(1)} = \frac{\sum_{t=1}^{T} \lambda_{m,t} (y_t - \mu_m^{(1)})^2}{\sum_{t=1}^{T} \lambda_{m,t}}$$

REPEAT

Multimodal Interaction Lab

b 3: m ɪ ŋ ə m

EE4R Automatic Spoken Language Processing

UNIVERSITY OF BIRMINGHAM

Calculation of $\lambda_{m,t}$

- In other words, $\lambda_{m,t}$ is the probability of the m^{th} component given the data point y_t
- From Bayes' theorem

Calculate from m^{th} Gaussian component

m^{th} weight

$$\lambda_{m,t} = P(m \mid y_t) = \frac{p(y_t \mid m) P(m)}{p(y_t)} = \frac{\boxed{p_m(y_t)}\,\boxed{w_m}}{\boxed{\sum_{k=1}^{M} p_k(y_t) w_k}}$$

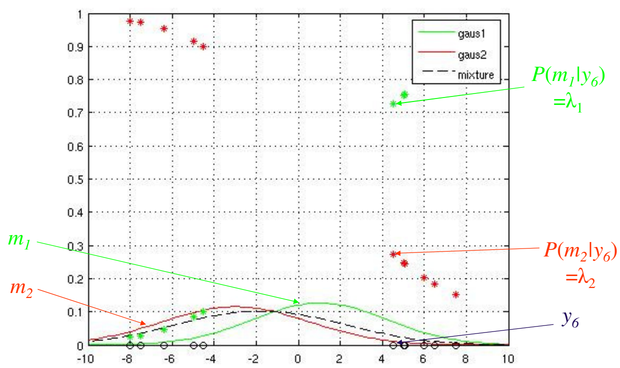
Sum over all components

Multimodal Interaction Lab

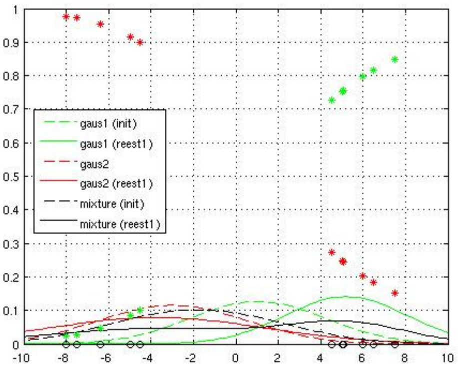
b 3: m ɪ ŋ ə m

UNIVERSITY OF BIRMINGHAM

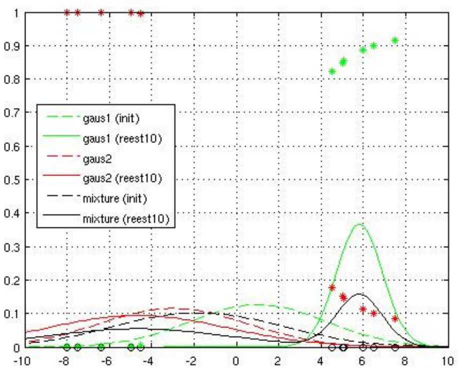
Example – initial model



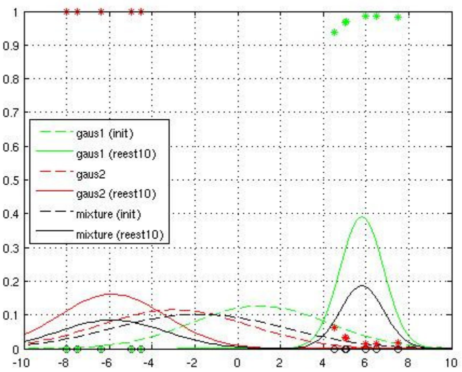
Example – after 1st iteration of E-M



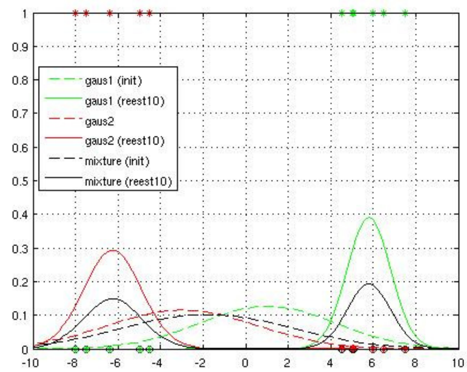
Example – after 2nd iteration of E-M



Example – after 4th iteration of E-M



Example – after 10th iteration of E-M



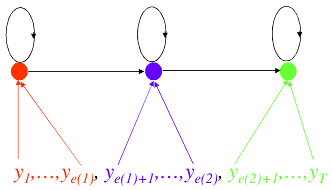
ML training for HMMs

- Now consider
  - An N state HMM M, each of whose states is associated with a Gaussian PDF
  - A training sequence $y_1,\ldots,y_T$
- For simplicity assume that each y_t is 1-dimensional

ML training for HMMs

- If we knew that:
  - $y_1,\ldots,y_{e(1)}$ correspond to state 1
  - $y_{e(1)+1},\ldots,y_{e(2)}$ correspond to state 2
  - :
  - $y_{e(n-1)+1},\ldots,y_{e(n)}$ correspond to state n
  - :
- Then we could set the mean of state n to the average value of $y_{e(n-1)+1},\ldots,y_{e(n)}$

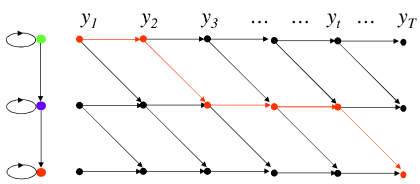
ML Training for HMMs



Unfortunately we <u>don't</u> know that $y_{e(n-1)+1},\ldots,y_{e(n)}$ correspond to state n…

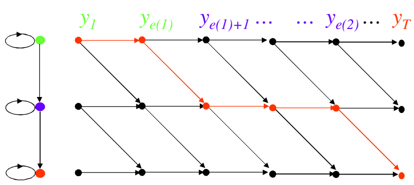
Solution

1. Define an initial HMM – M_0
2. Use the Viterbi algorithm to compute the optimal state sequence between M_0 and $y_1,\ldots,y_T$



Solution (continued)

- Use optimal state sequence to segment y



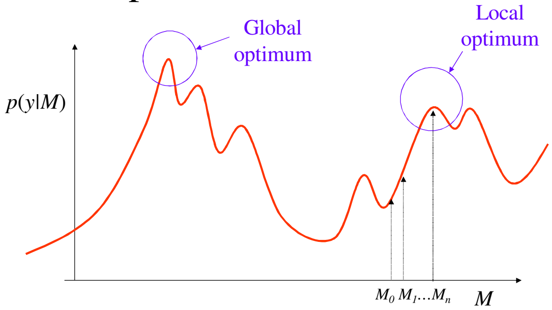
- Reestimate parameters to get a new model M_1

Solution (continued)

- Now repeat whole process using M_1 instead of M_0, to get a new model M_2
- Then repeat again using M_2 to get a new model M_3
- ….

$p(y \mid M_0) \leq p(y \mid M_1) \leq p(y \mid M_2) \leq \ldots \leq p(y \mid M_n) \ldots$

Local optimization



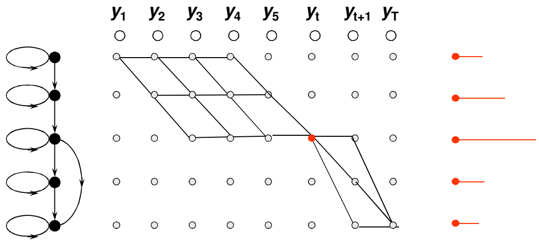
Baum-Welch optimization

- The algorithm just described is often called Viterbi training or Viterbi reestimation
- It is often used to train large sets of HMMs
- An alternative method is called Baum-Welch reestimation – it is a soft version of the Viterbi estimation
- Reestimation of mean value associated with state i:

$$\mu(i) = \frac{\sum_{t=1}^{T} \gamma_t(i) y_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

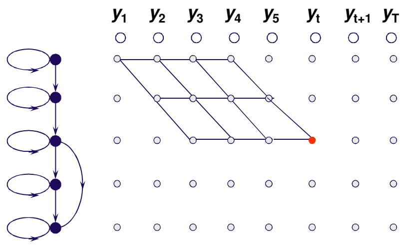
Baum-Welch Reestimation

$P(x_t=i|Y)=\gamma_t(i)$



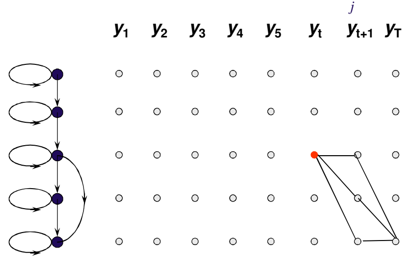
‘Forward’ Probabilities

$$\alpha_t(i) = \text{Prob}(y_1,...,y_t \text{ and } x_t = i \mid M) = \sum_j \alpha_{t-1}(j)a_{ji}b_i(y_t)$$



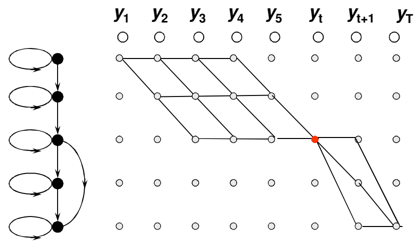
‘Backward’ Probabilities

$$\beta_t(i) = \text{Prob}(y_{t+1},...,y_T \mid x_t = i, M) = \sum_j a_{ij}\beta_{t+1}(j)b_j(y_{t+1})$$



'Forward-Backward' Algorithm

$$\gamma_t(i) = P(x_t = i \mid Y) = \frac{P(Y, x_t = i)}{P(Y)} = \frac{P(Y, x_t = i)}{\sum_{i=1}^{N} P(Y, x_t = i)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$



$$\mu(i) = \frac{\sum_{t=1}^{T} \gamma_t(i) y_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

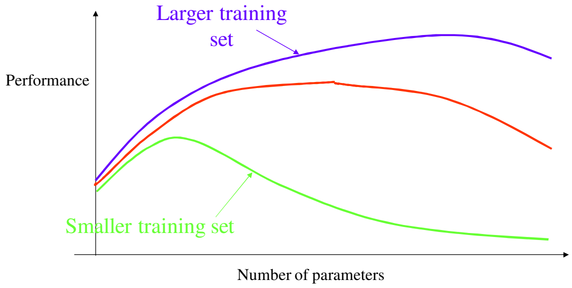
Adaptation

- A modern large-vocabulary continuous speech recognition system has <u>many thousands of parameters</u>
- Many hours of speech data used to train the system (e.g. 200+ hours!)
- Speech data comes from many speakers
- Hence recogniser is 'speaker independent'
- But performance for an individual would be better if the system were <u>speaker dependent</u>

Adaptation

- For a single speaker, only a small amount of training data is available
- Viterbi reestimation or Baum-Welch reestimation <u>will not work</u>
- Adaptation:
  - the problem of robustly adapting a <u>large</u> number of model parameters using a <u>small</u> amount of training data

‘Parameters vs training data’



Adaptation

- Two common approaches to adaptation (with small amounts of training data)
  - Bayesian adaptation (also known as MAP adaptation (MAP = Maximum a Posteriori))
  - Transform-based adaptation (also known as MLLR (MLLR = Maximum Likelihood Linear Regression))

Bayesian (MAP) adaptation

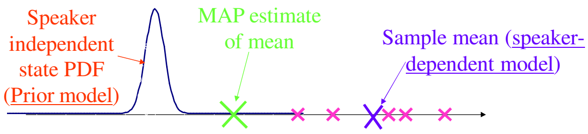
- MAP estimation maximises the posterior probability of M given the data y, i.e., $P(M \mid y)$
- From Bayes’ Theorem:

$$P(M \mid y) = \frac{p(y \mid M)P(M)}{p(y)}$$

- $P(M)$ is the prior probability of M
- $p(y \mid M)$ is the likelihood of the adaptation data on M

Bayesian (MAP) adaptation

- Uses well-trained, ‘speaker-independent’ HMM as a prior *P(M)* for the estimate of the parameters of the speaker dependent HMM
- E.G:



Bayesian (MAP) adaptation

$$\hat{M} = \lambda M_{prior} + (1-\lambda) M_y, \; 0 \le \lambda \le 1$$

MAP model — Prior model — ‘Speaker-dependent’ model

- Intuitively, if the adaptation data set *y* is big, then the MAP adapted model will be biased towards *y*, so λ will be small
- Conversely, if there is very little adaptation data, the MAP model will be biased towards the prior, so λ will be big

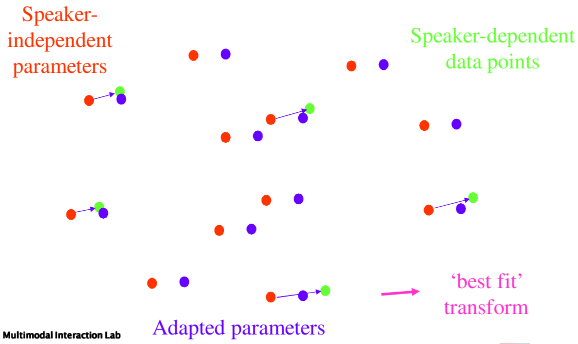
Transform-based adaptation (MLLR)

- Maximum Likelihood Linear Regression (MLLR) is another method for adapting the mean vectors of a set of HMMs
- Estimate a linear transform to transform speaker-independent into speaker-dependent parameters
- Suppose that M_{SI} is a speaker-independent HMM with Gaussian Mixture state output PDFs
- Suppose *A* is linear transformation on the *D*-dimensional space of acoustic vectors and that *b* is an acoustic vector
- Let $M_{SD} = T(M_{SI})$ be the HMM derived from M_{SI} by replacing each Gaussian mean vector μ with $A\mu + b$

MLLR adaptation

- Given data y from a new speaker, the aim of <u>MLLR</u> is to find A and b such that $P(y|T(M_{SI}))$ is maximised
- … hence <u>Maximum Likelihood</u> LR
- Need to estimate the $D \times D$ parameters of A
- Each acoustic vector is typically 40 dimensional, so a <u>linear transform</u> of the acoustic data needs 40*40 = 1600 parameters
- This is much less than the 10s or 100s of thousands of parameters needed to train the whole system
- Same transformation A can be used for all models and states.
- Alternatively, if there is enough data from the new speaker, a separate transformation can be estimated for each model, state, or set of states

Transform-based adaptation



Summary

- Maximum Likelihood (ML) estimation
- Parameter estimation for GMM
- Viterbi HMM parameter estimation
- Baum-Welch HMM parameter estimation
- Forward and backward probabilities
- Adaptation: –Bayesian (MAP); –Transform-based (MLLR)
  - J-L Gauvain and C-H Lee, "Bayesian learning for Hidden Markov Models with Gaussian mixture state observation densities", *Speech Communication* 11, pp 205-213, 1992
  - C J Leggeter and P C Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density HMMs", *Computer Speech and Language,* 9, pp 171-186, 1995