Training HMMs

Outline

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

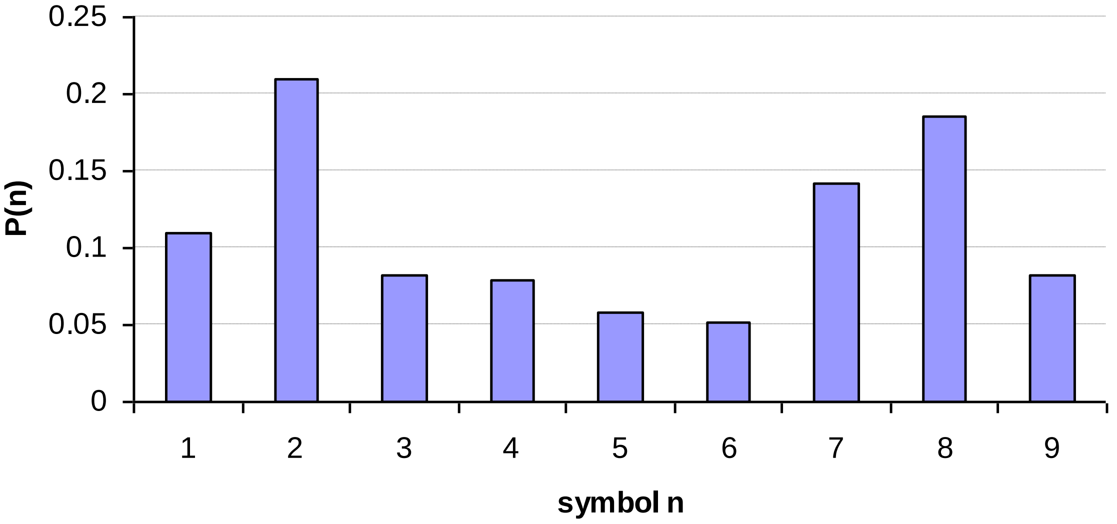
Discrete variables

- 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:

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

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

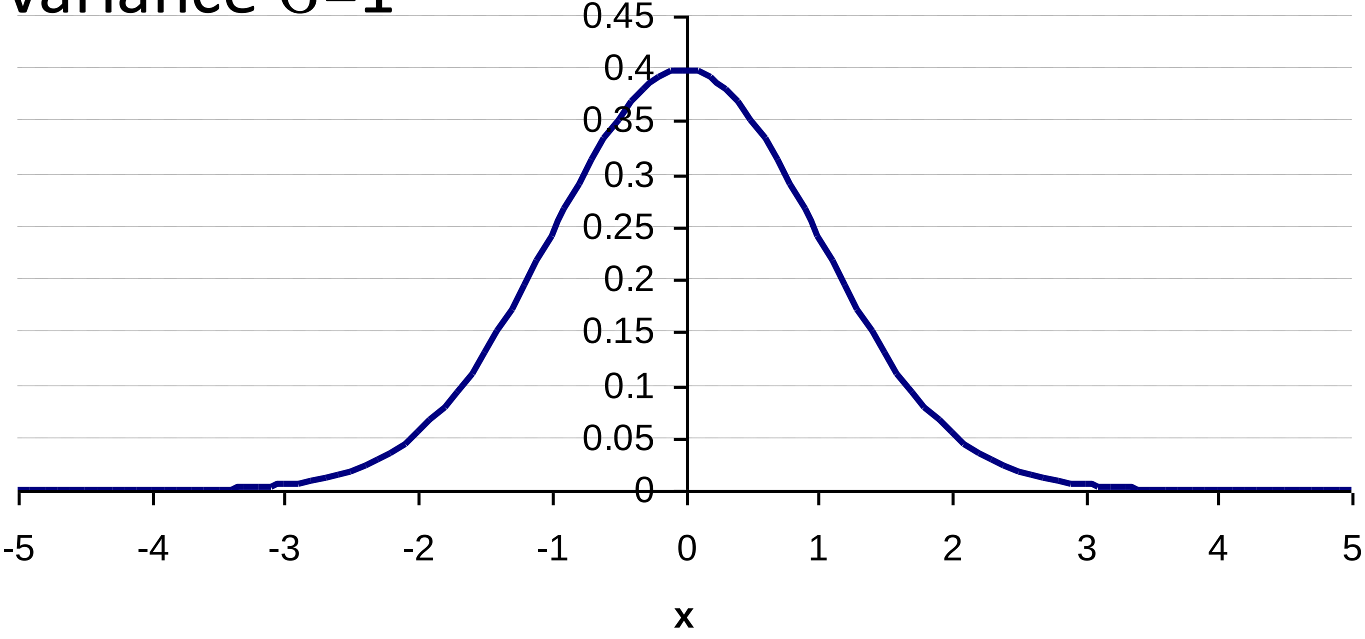
- In most practical applications the data are not restricted to a finite set of values – they can take any value in *N*-dimensional space
- Simply counting the number of occurrences of each value is no longer a viable way of estimating probabilities…
- …but there are generalisations of this approach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

Continuous Random Variables

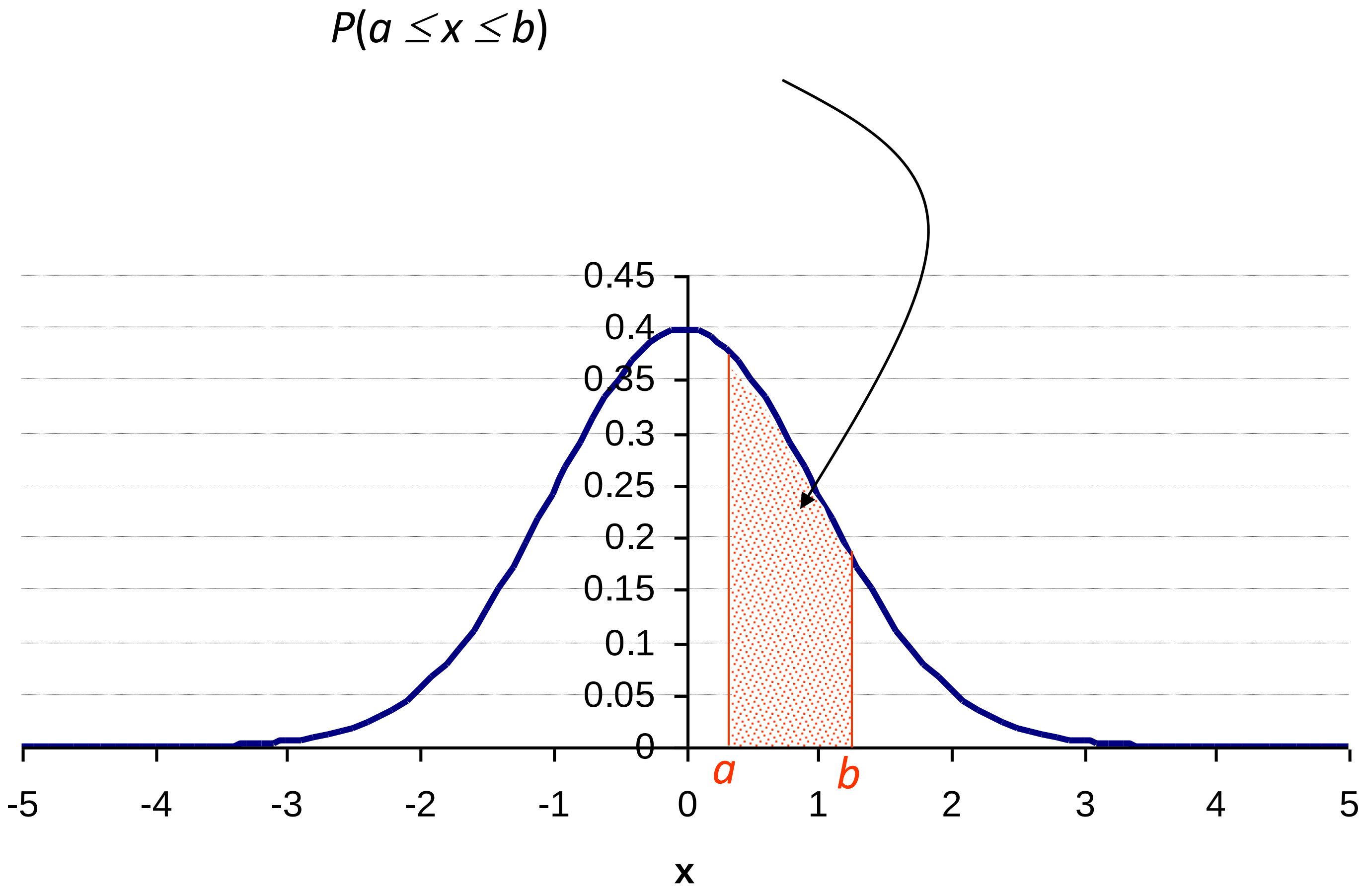
- An alternative is to use a parametric model
- In a parametric model, probabilities are defined by a small set of parameters
- Simplest example is a normal, or Gaussian model
- A Gaussian probability density function (PDF) is defined by two parameters
  - its mean μ, and
  - variance σ

Gaussian PDF

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



Gaussian PDF



Gaussian PDF

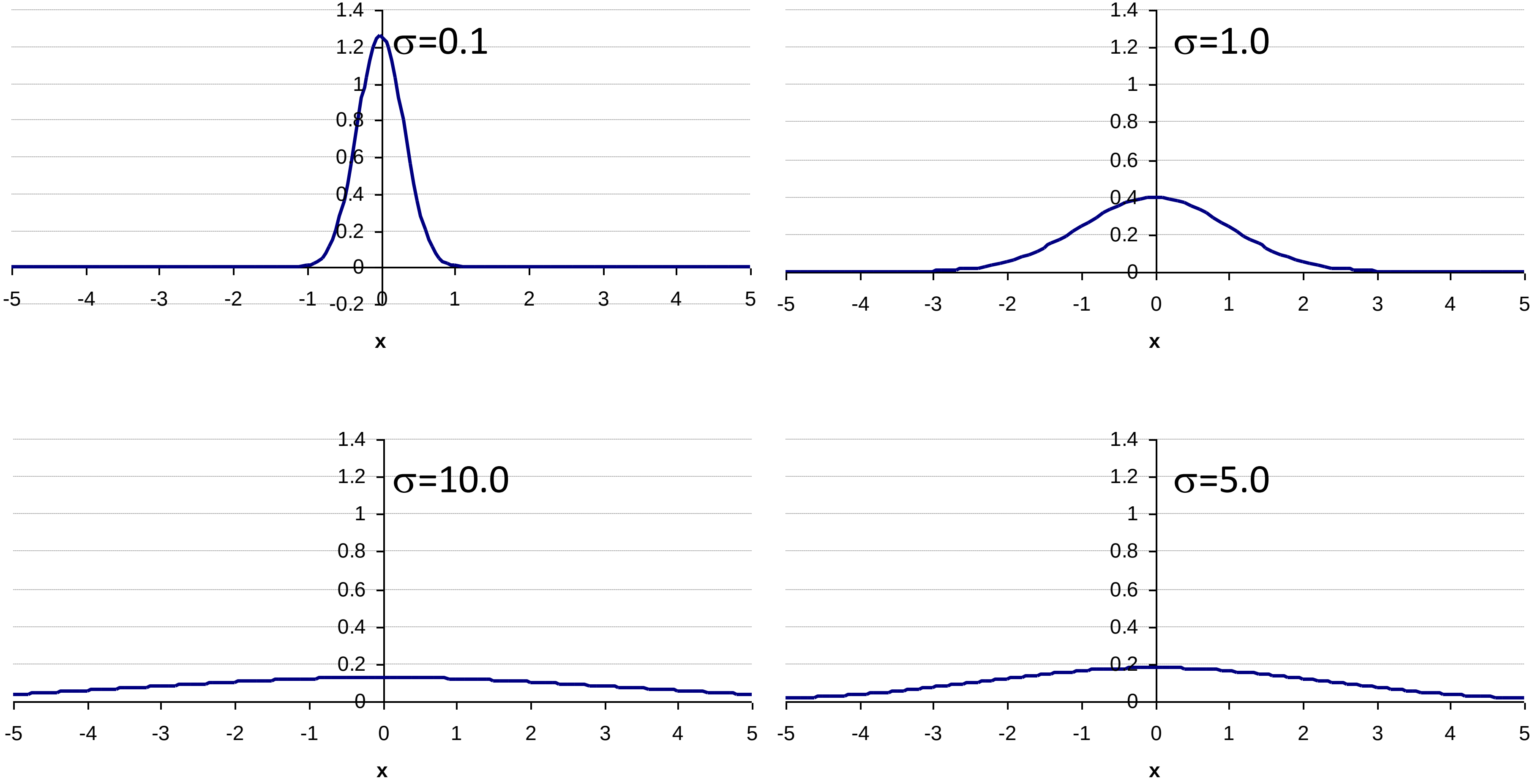
- For a 1-dimensional Gaussian PDF *p* with mean μ and variance σ:

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

Constant to ensure area under curve is 1

Defines ‘bell’ shape

More examples



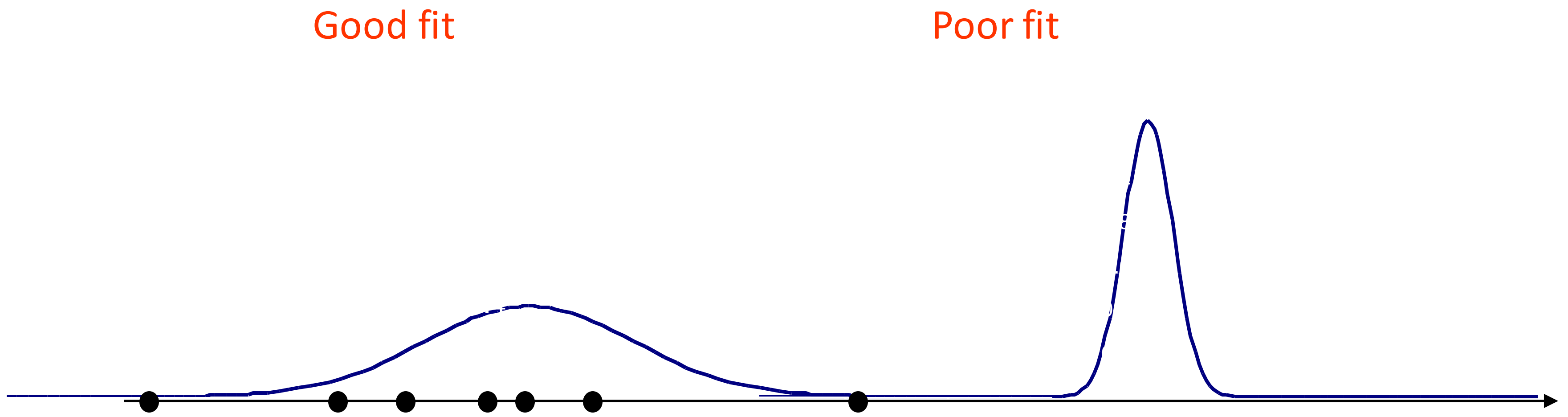
Fitting a Gaussian PDF to Data

- 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,

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

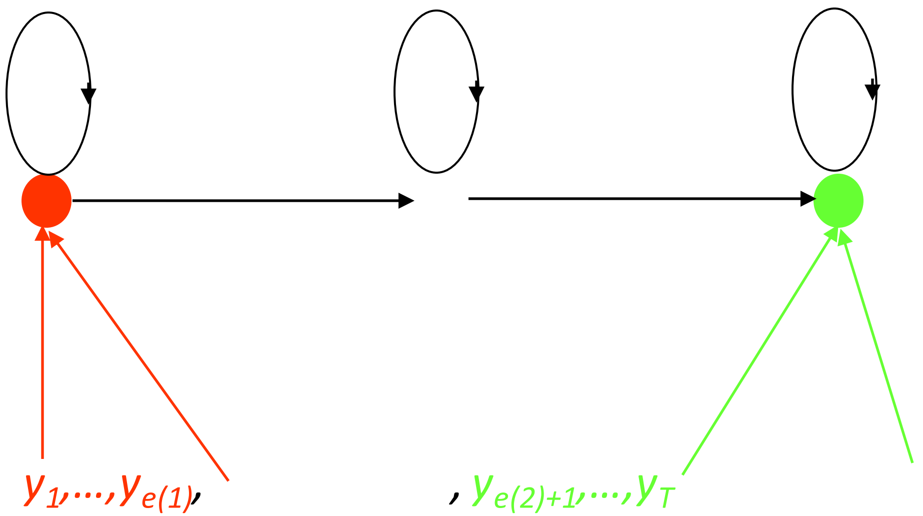
ML training for HMMs

- Now consider
  - An N state HMM M, each of whose states is associated with a <u>Gaussian</u> 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 $y_{e(n-1)+1},\ldots,y_{e(n)}$

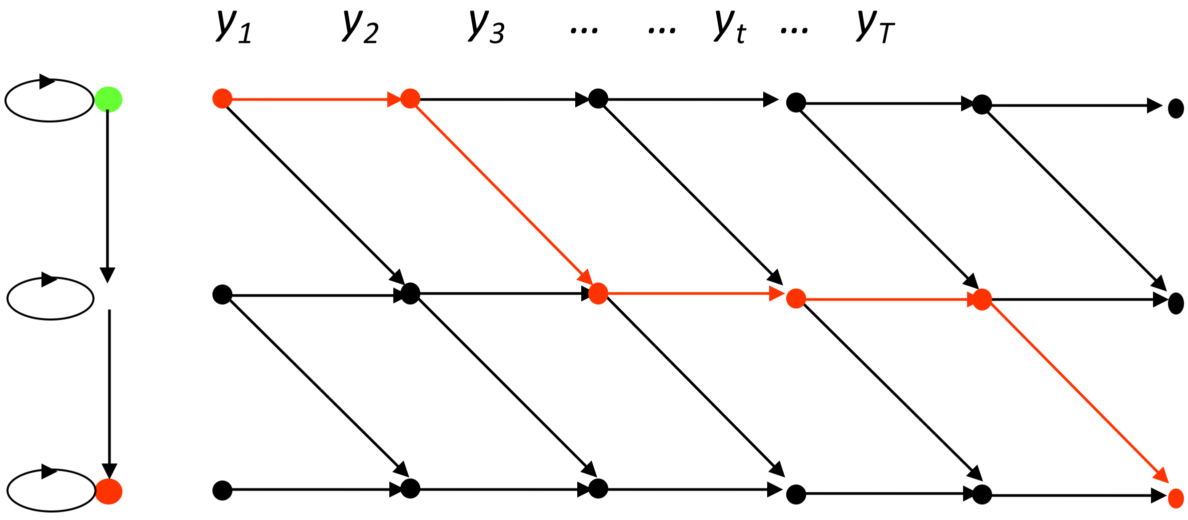
ML Training for HMMs



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

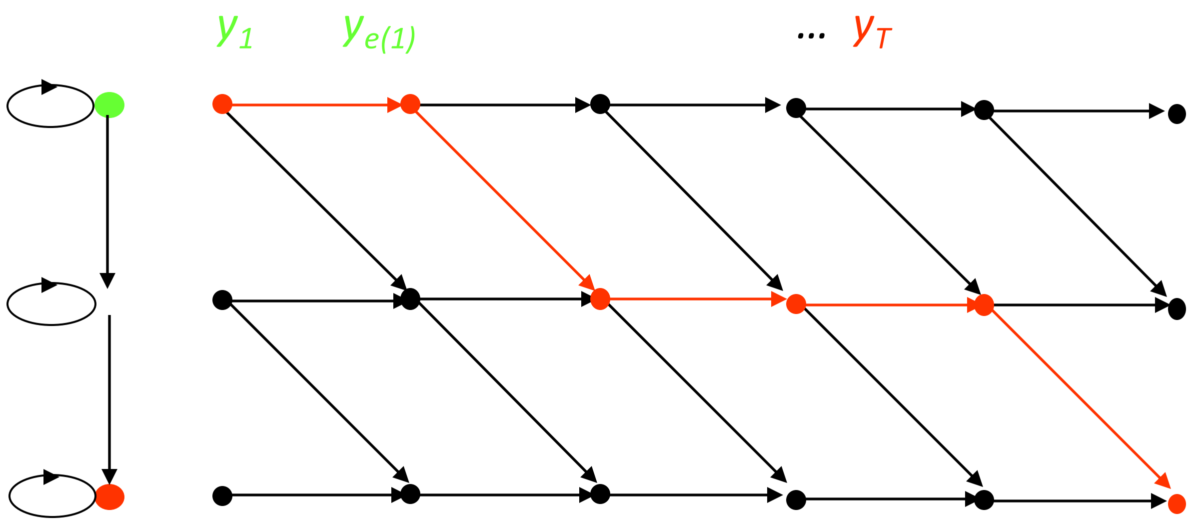
Solution

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*



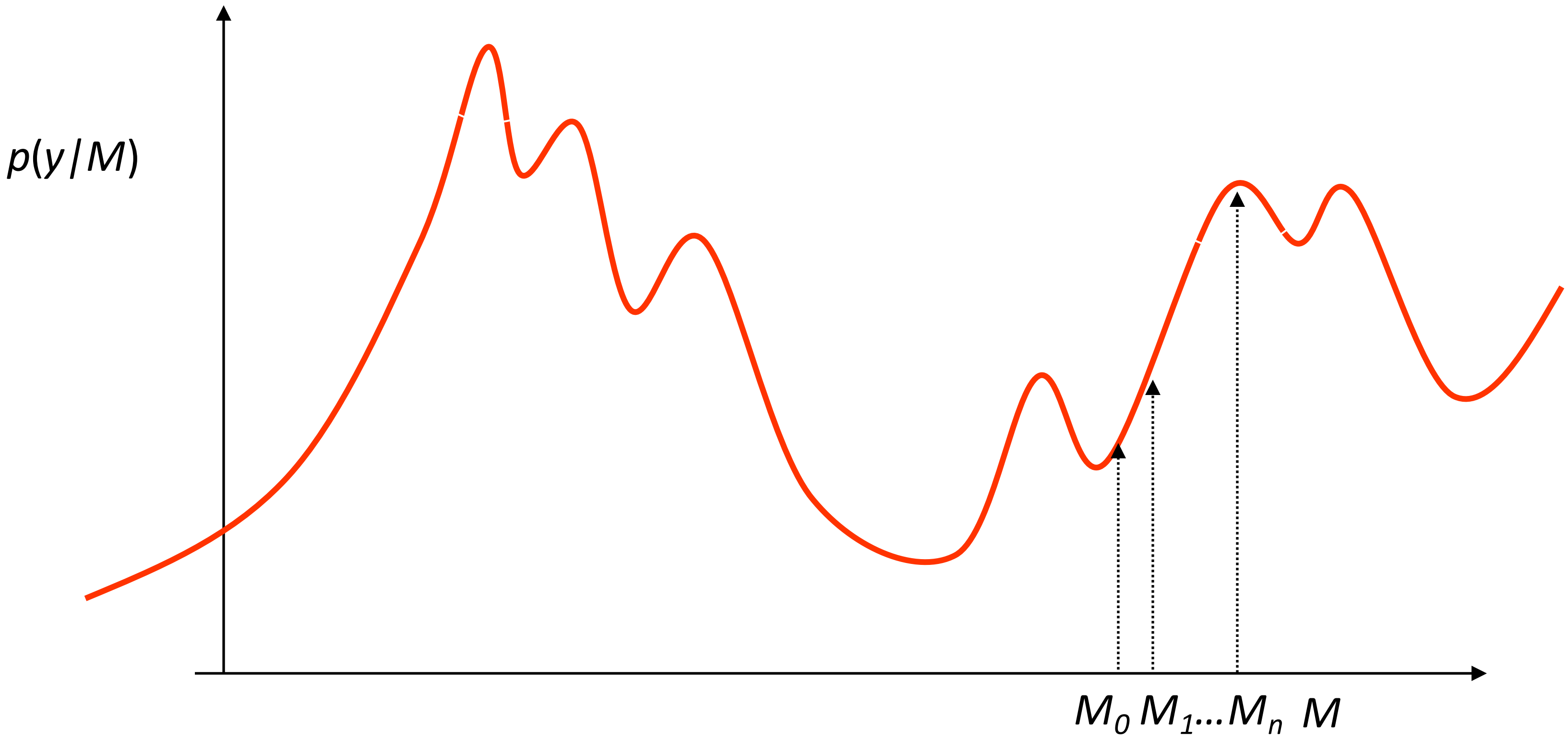
- Reestimate parameters to get a new model M_1

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

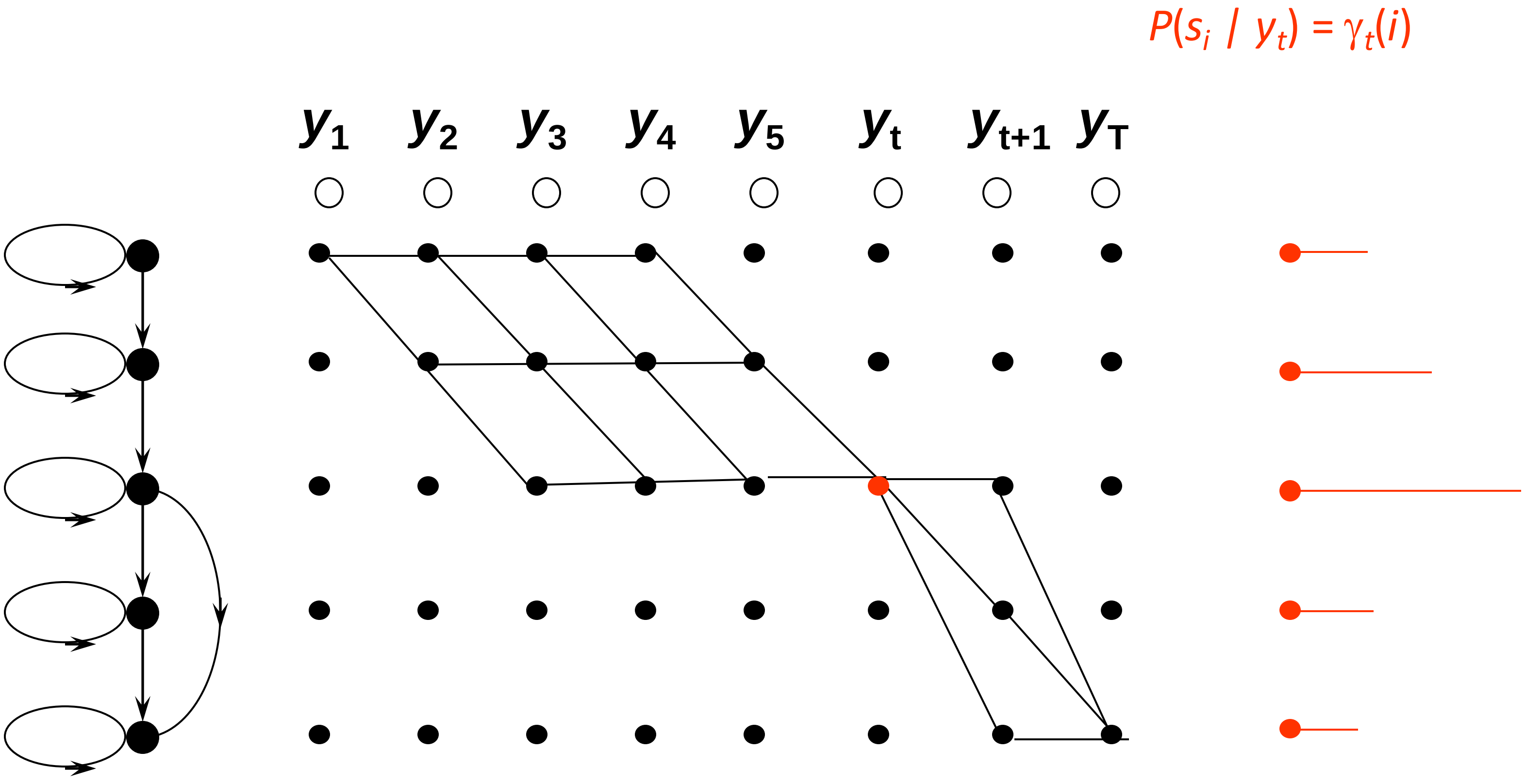


Baum-Welch optimization

- 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

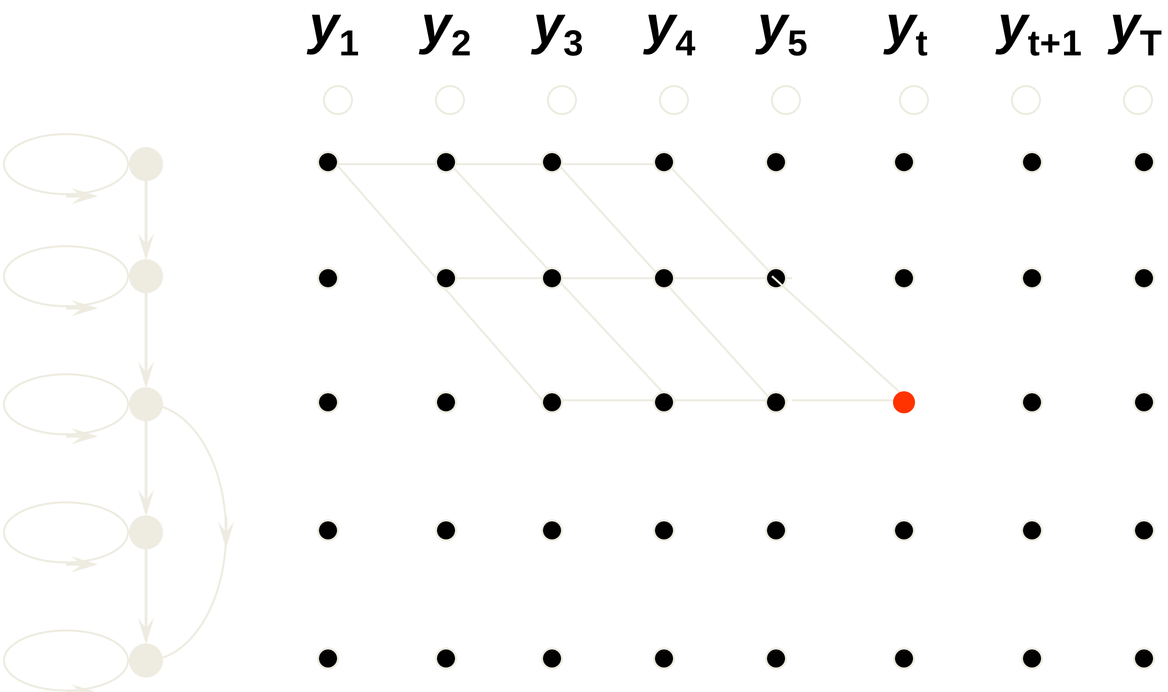
$$\mu(i) = \frac{\sum_{t=1}^{T}\sum_{X\in S_{i,t}} P(Y,X|M_0)\, y_t}{\sum_{t=1}^{T}\sum_{X\in S_{i,t}} P(Y,X|M_0)} = \sum_{t=1}^{T}\gamma_t(i)\, y_t$$

Baum-Welch Reestimation



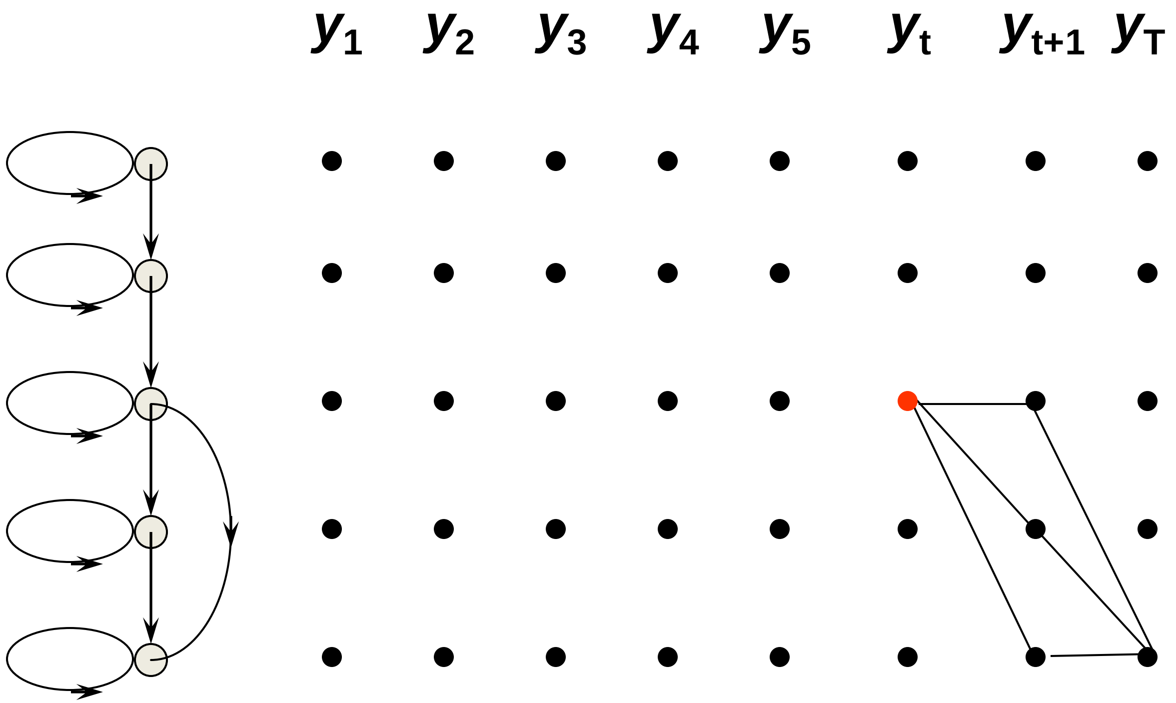
'Forward' Probabilities

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

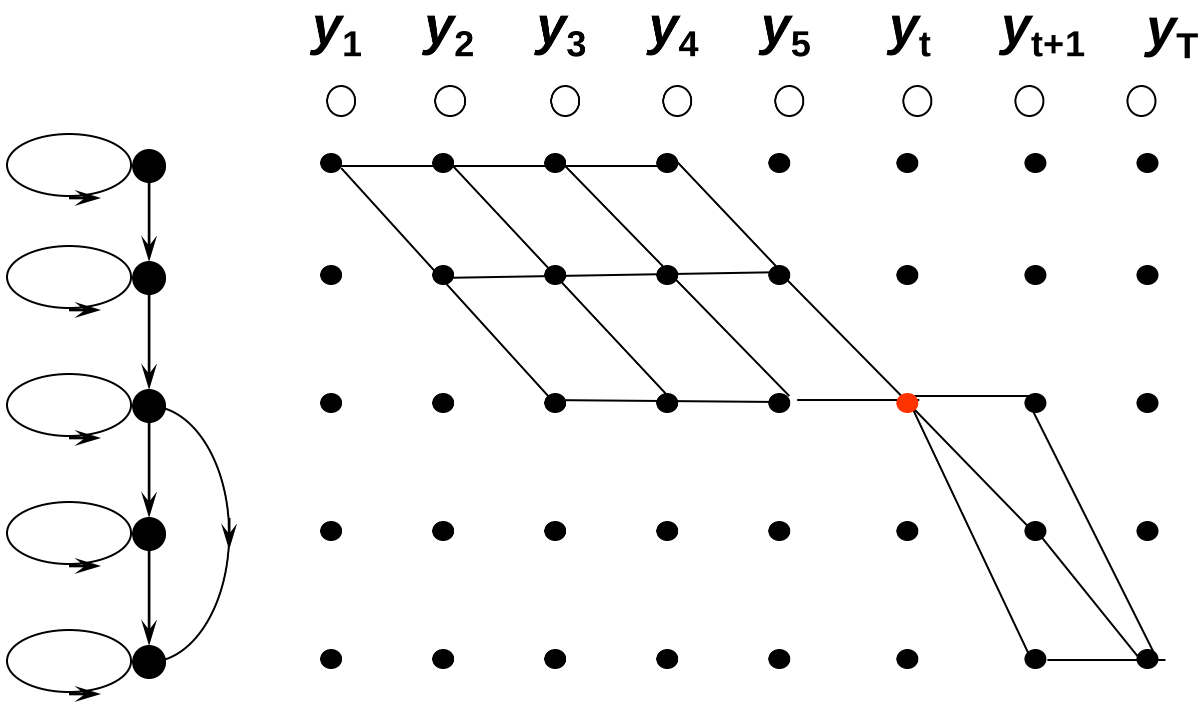


‘Backward’ Probabilities

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



'Forward-Backward' Algorithm



$$\gamma_t(i) = \frac{\bar{\gamma}_t(i)}{\sum_{t=1}^{T} \bar{\gamma}_t(i)}$$

$$\bar{\gamma}_t(i) = \alpha_t(i)\beta_t(i) \qquad \mu(i) = \sum_{t=1}^{T} \gamma_t(i) y_t$$

'Forward-Backward' Algorithm

$$\gamma_t(i) = \frac{\alpha_t(i)\ \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\ \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\ \beta_t(i)}$$

$$\widehat{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_i(t) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\widehat{\Sigma}_i = \frac{\sum_{t=1}^{T} \gamma_i(t)(\boldsymbol{o}_t - \widehat{\mu}_i)(\boldsymbol{o}_t - \widehat{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_i(t)}$$

These are weighted averages $\Rightarrow$ weighted by Prob. of being in state j at t

Re-estimate Transition Probabilites

$$\xi_t(i, j) = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}$$

$\bar{\pi}_i$ = expected frequency (number of times) in state S_i at time $(t = 1) = \gamma_1(i)$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

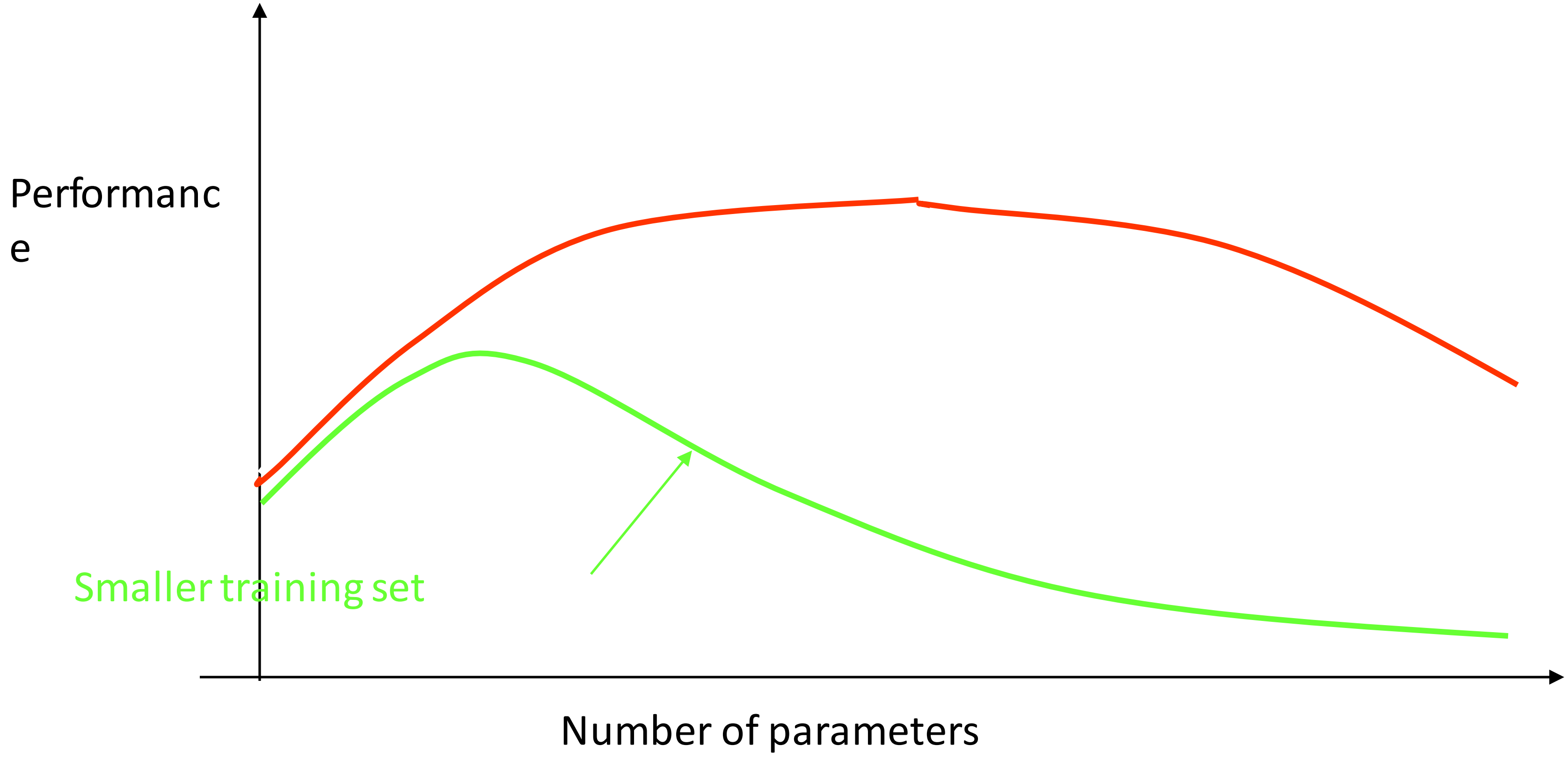
Adaptation

- A modern large-vocabulary continuous speech recognition system has many thousands of parameters
- 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 speaker dependent

Adaptation

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

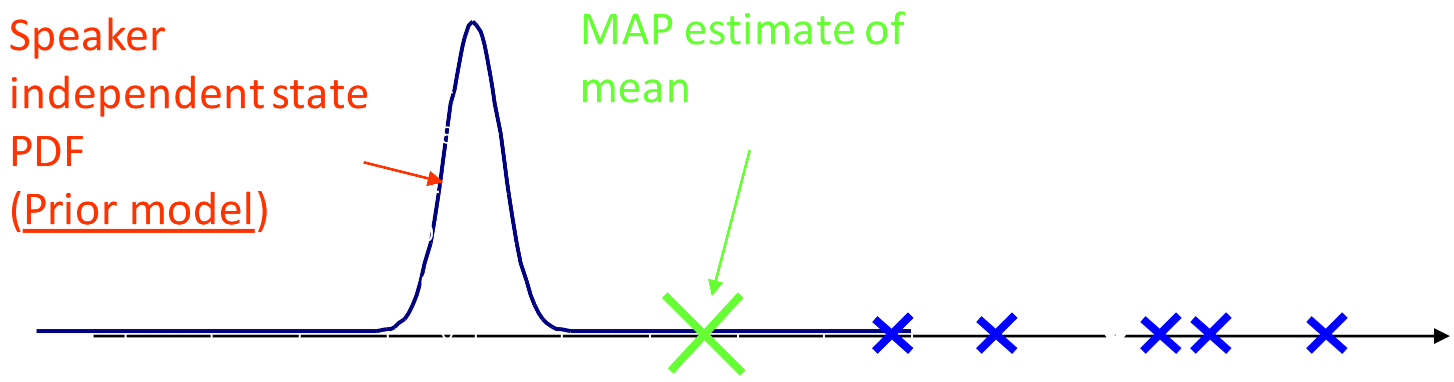
- MAP estimation maximises the posterior probability of *M* given the data *y*, i.e., $P(M \mid y)$
- From Bayes' Theorem:

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

- $P(M)$ 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