Data Mining and Machine Learning

HMM Tra

Peter Jančovič

Objectives

- Reminder: Maximum Likelihood (ML) parameter estimation
  - ML for Gaussian PDFs and for GMMs
- ML for HM
  - Viterbi-style training
  - Baum-Welch algorithm

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?
- Define the best fitting Gaussian to be the one such that $p(y|\mu,\sigma)$ is maximised – 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: the likelihood is maximised by setting:

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

ML training for GMMs

- Now consider
  - A Gaussian Mixture Model with M components has
    - M means: $\mu_1, \ldots, \mu_M$
    - M varia
    - M mixtu $w_1,$
  - A training sequence $y_1, \ldots, y$
- 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 th[th]
  - Set σ_m to b[les which belong to the m^{th} component
  - Set w_m to be the proportion [hich 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

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

1. For each m ties

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

2. Use these probabilities to work out 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

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

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 steps 1-3

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

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

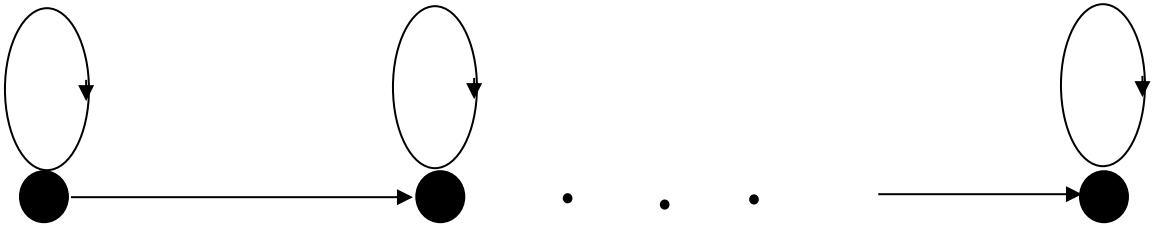
rom m^{th} component

m^{th} weight

Sum over all components

ML training for HMMs

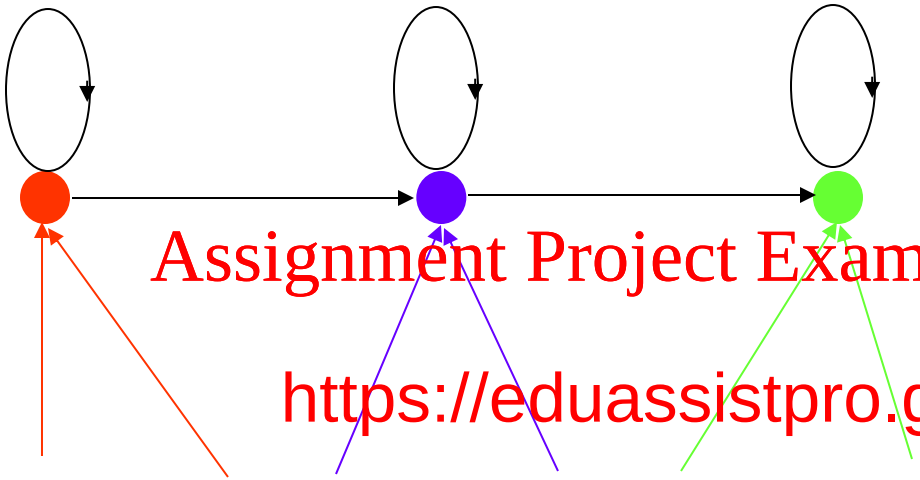
- Now consider
  - An N state HMM M, each of whose states is associated with a Gaussian PDF
  - A training s
- For simplicit -dimensional



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
  - :
  - $y_{e(n-1)+1}, \ldots, y$
  - :
- Then we could set the mean of state n to the average value of $y_{e(n-1)+1}, \ldots, y_{e(n)}$

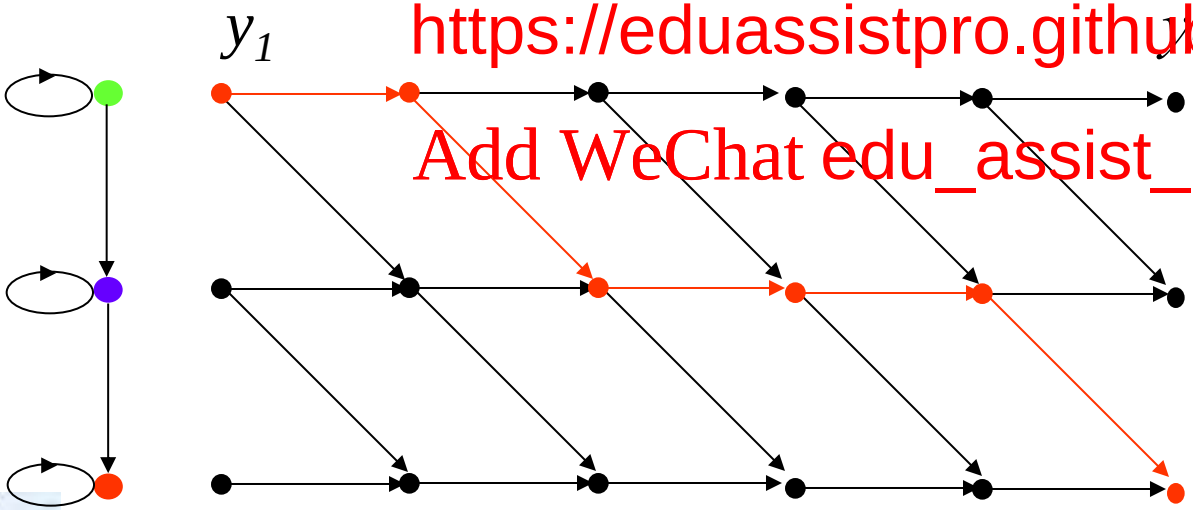
ML Training for HMMs



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

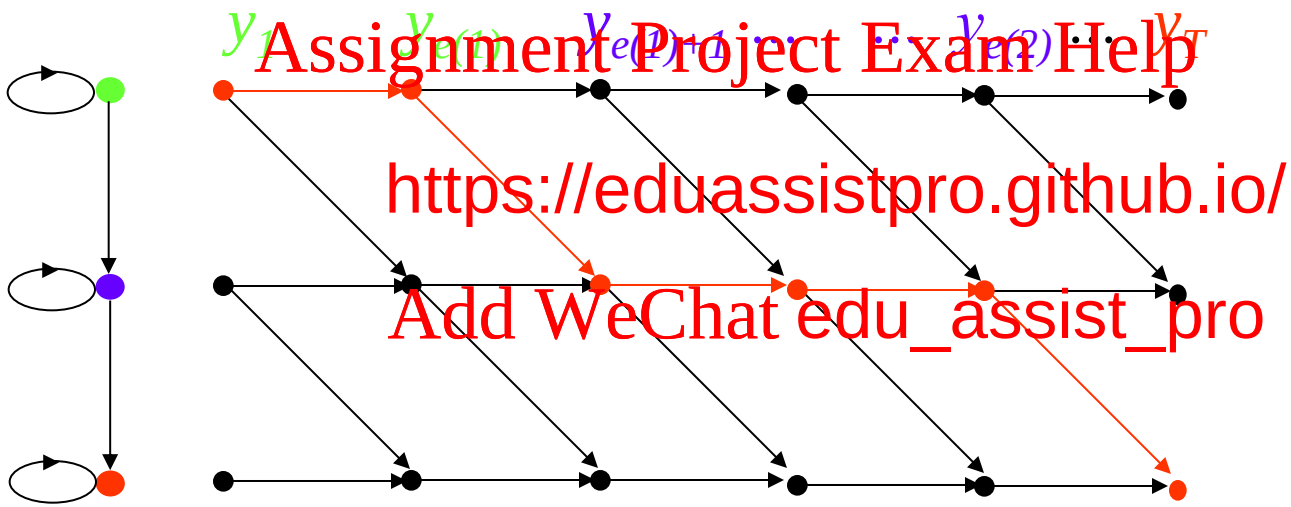
Solution

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



Solution (continued)

- Use optimal state sequence to segment y



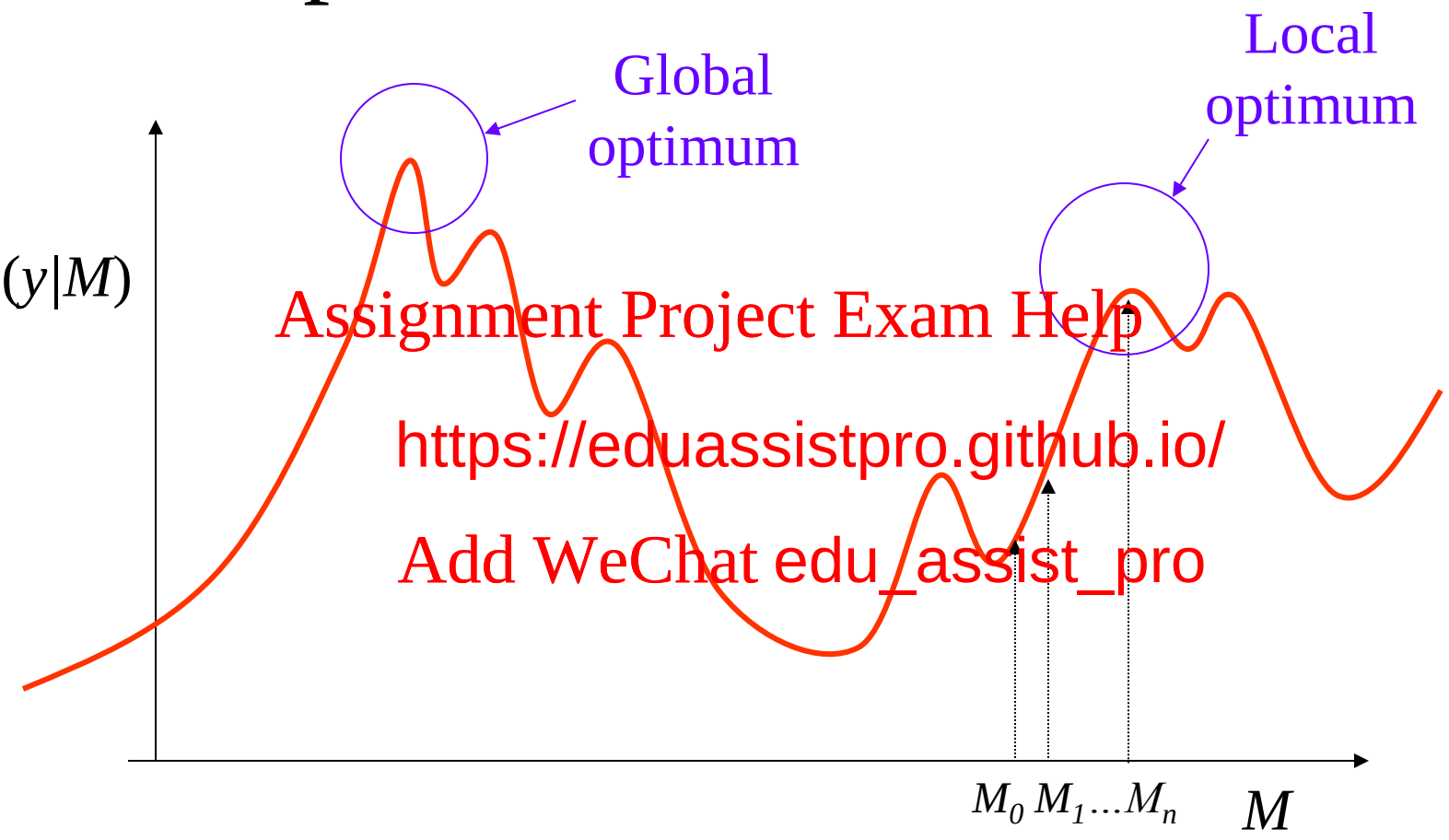
- Reestimate parameters to get a new model M_1

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



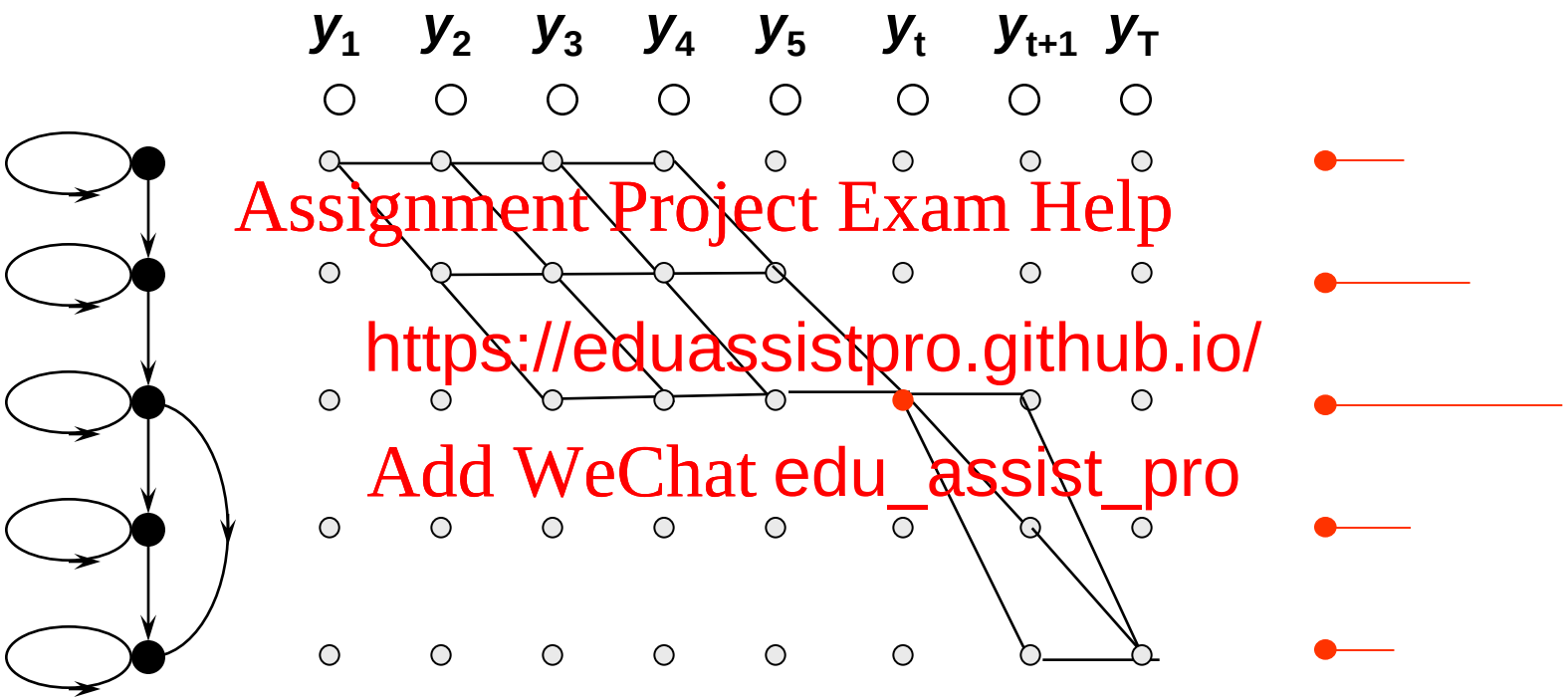
Baum-Welch optimization

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

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

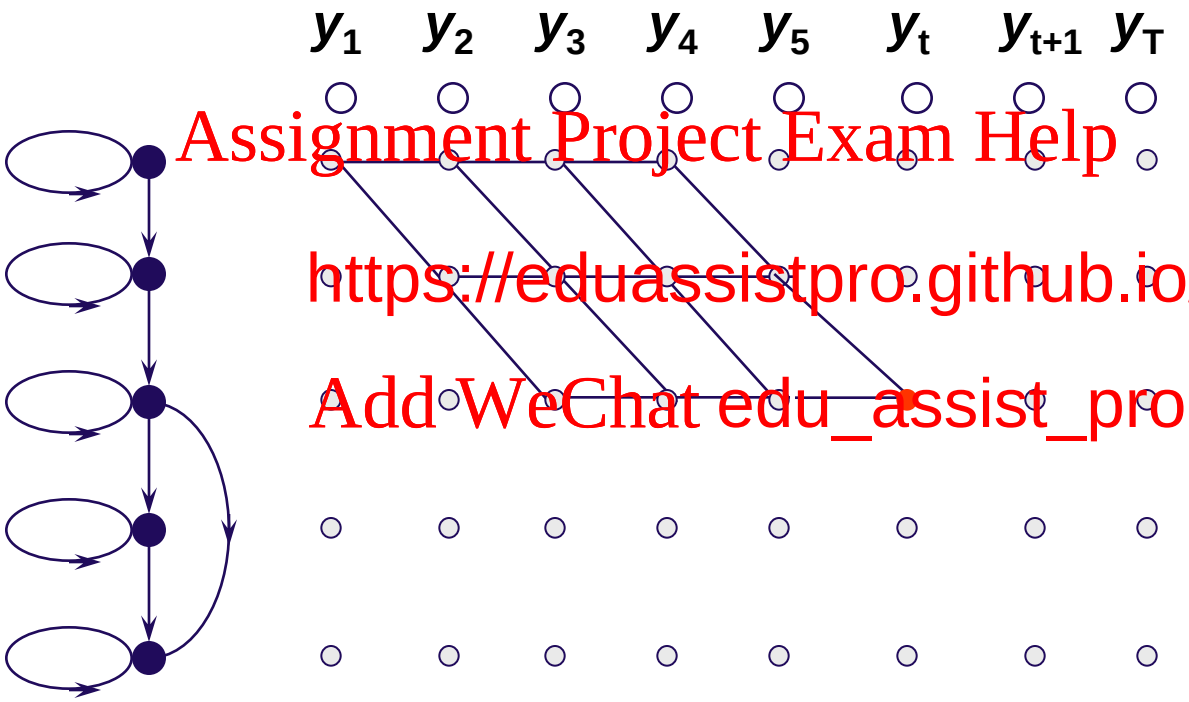
Baum-Welch Reestimation

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



‘Forward’ Probabilities

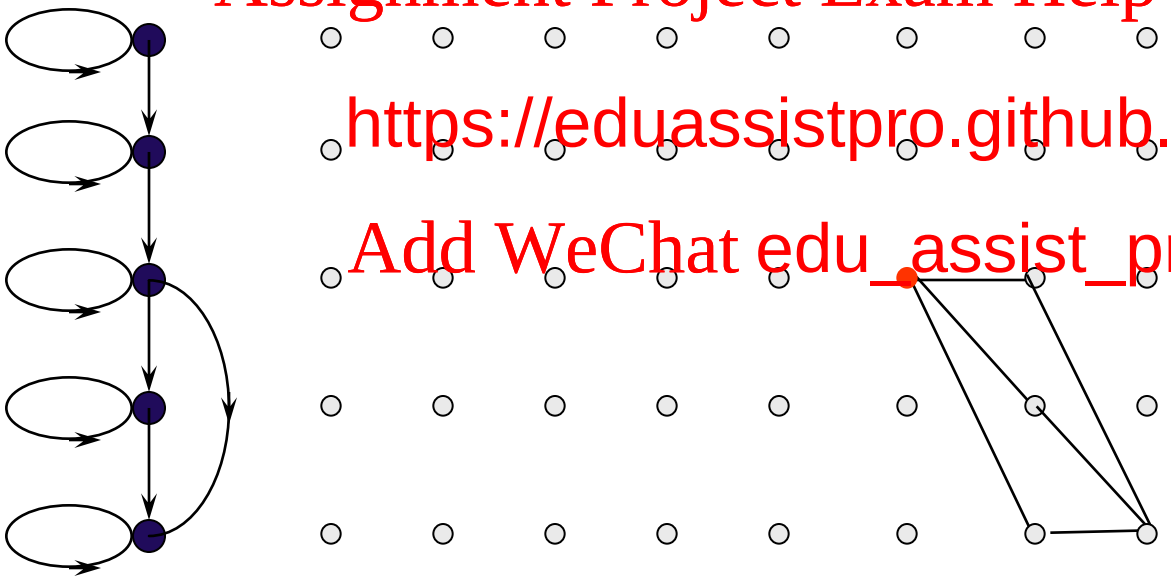
$$\alpha_t(i) = \text{Prob}(y_1,\ldots,y_t \text{ and } x_t = i \mid M) = \sum_j \alpha_{t-1}(j) a_{ji} b_i(y_t)$$



‘Backward’ Probabilities

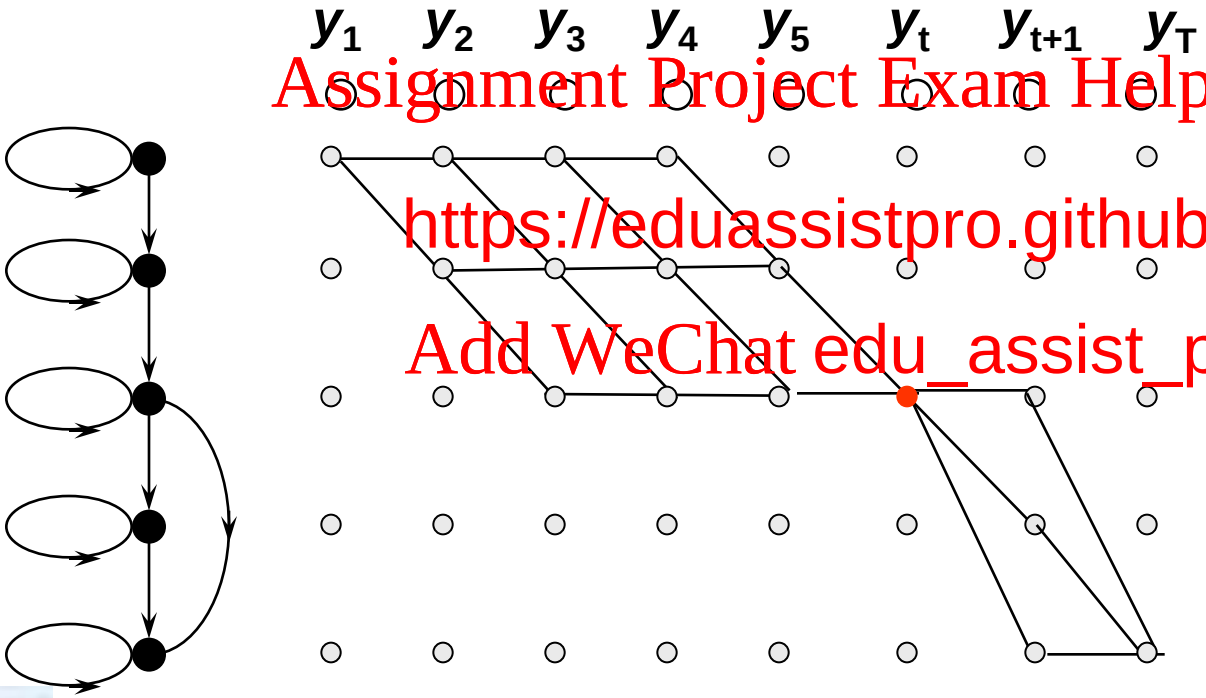
$$\beta_t(i) = \text{Prob}(y_{t+1},\ldots, y_T \mid x_t = i, M) = \sum_j a_{ij}\beta_{t+1}(j)b_j(y_{t+1})$$

y_1 y_2 y_3 y_4 y_5 y_t y_{t+1} y_T



‘Forward-Backward’ Algorithm

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



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

Notes on HMM parameter estimation

- The Baum-Welch/Viterbi algorithm is only guaranteed to find a **locally** optimal HMM set – hence choice of M_0 can be important
- Baum-Welch/Viterbi is a **supervised** training algorithm which requires
- The labelling el as the HMM set – phoneme level HMMs can be trained using data labelled orthographically at the phrase level
- For large applications B-W reestimation can be **very** computationally expensive

Summary

- Maximum Likelihood (ML) estimation
- Viterbi HMM
- Baum-Welch HMM parame on
  - Forward and backward probab