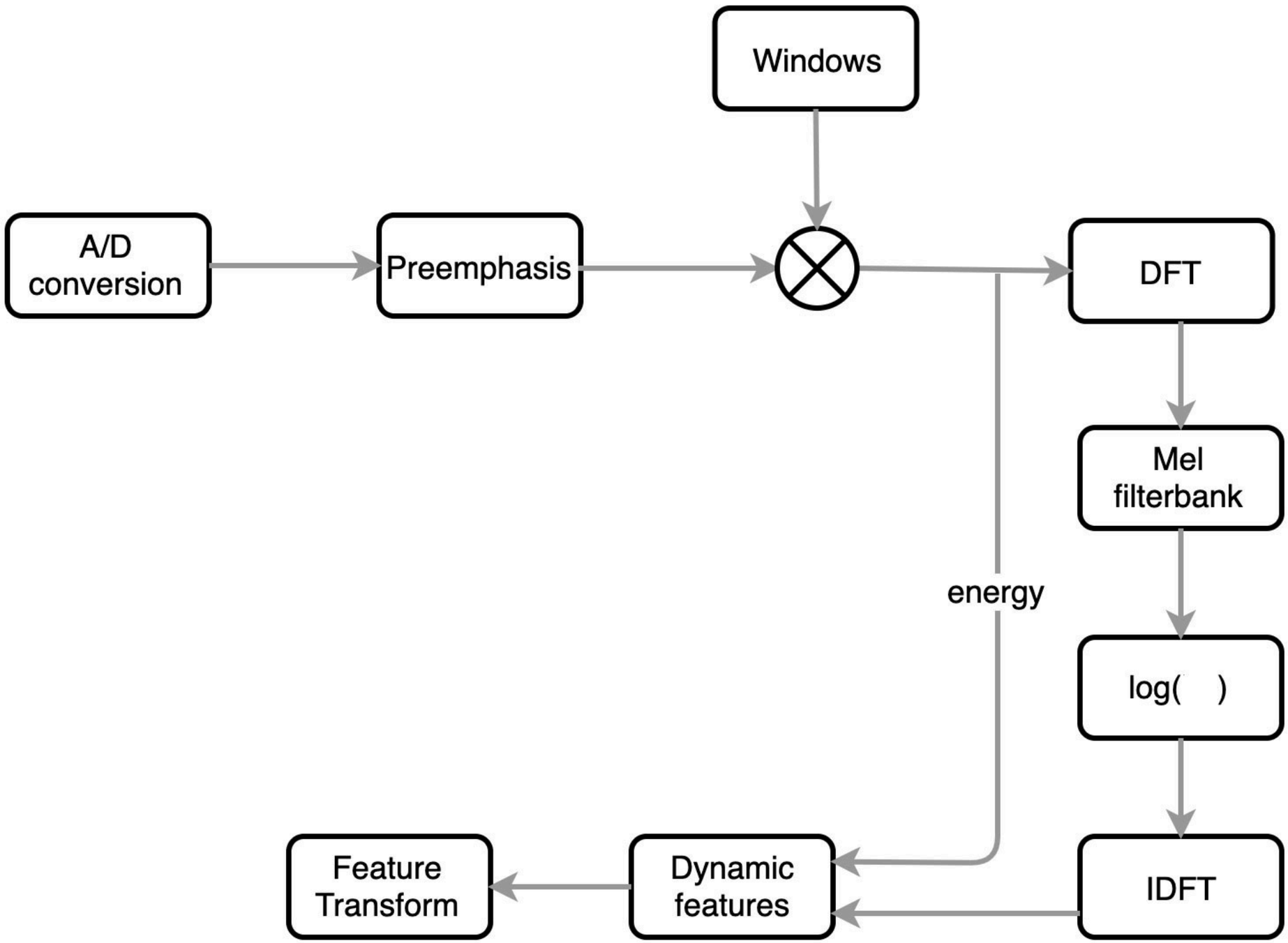
ASR Intro

GMM-HMM ASR

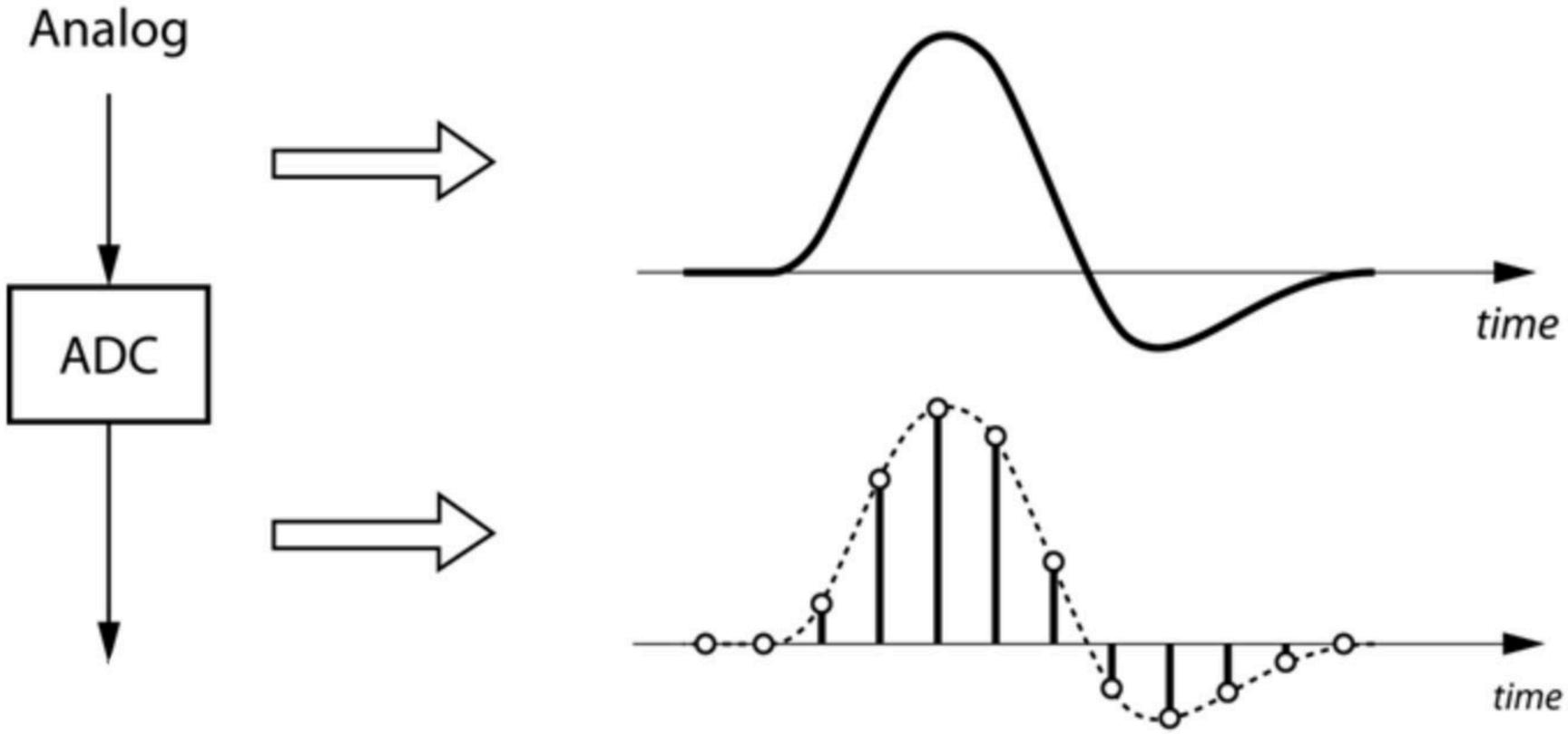
Plan

- Reminder: Features extraction
- ASR
- GMM
- HMM
- Viterbi Decoding
- Baum-Welch algorithm

MFCC

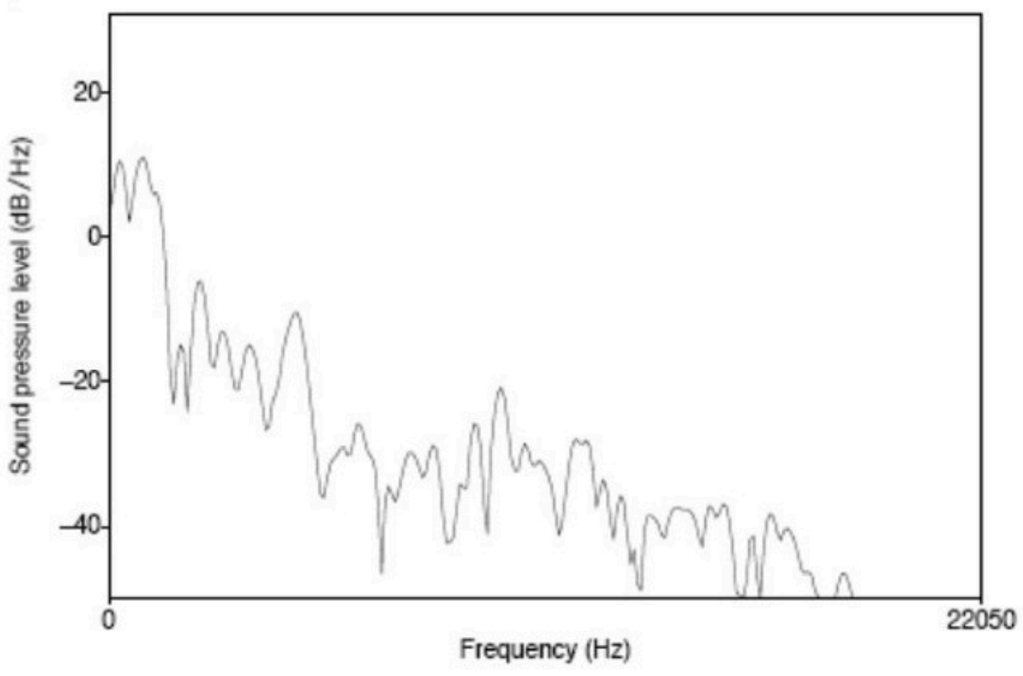

Windows
A/D conversion
Preemphasis
DFT
Mel filterbank
energy
log( )
Feature Transform
Dynamic features
IDFT

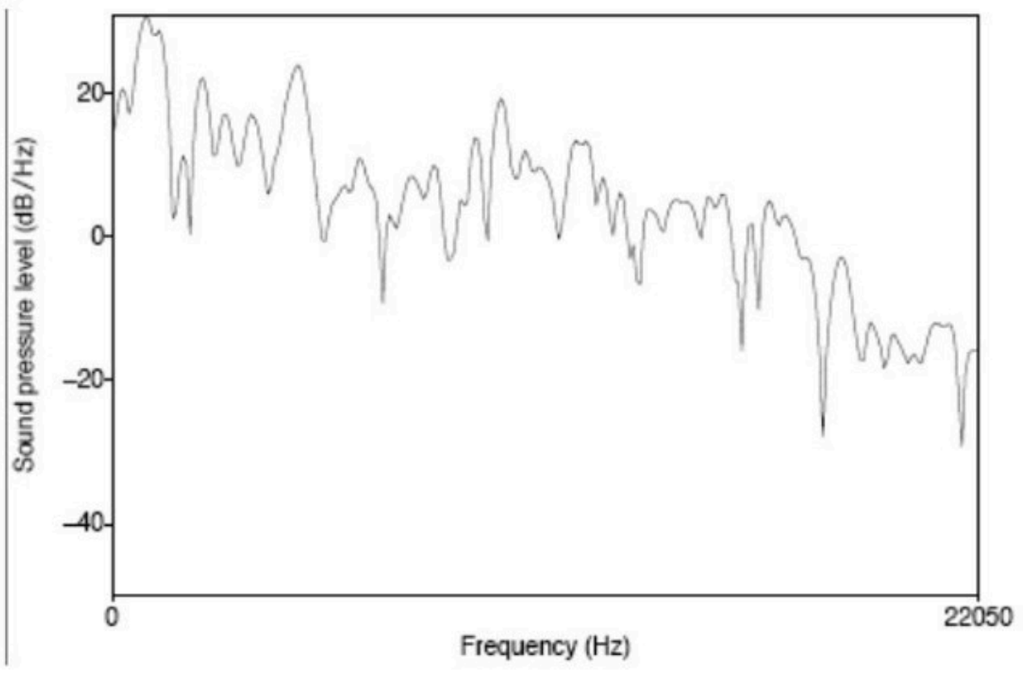
A/D Conversion



Pre-emphasis

$$x'[t_d] = x[t_d] - \alpha x[t_d - 1] \qquad 0.95 < \alpha < 0.99$$





Windowing

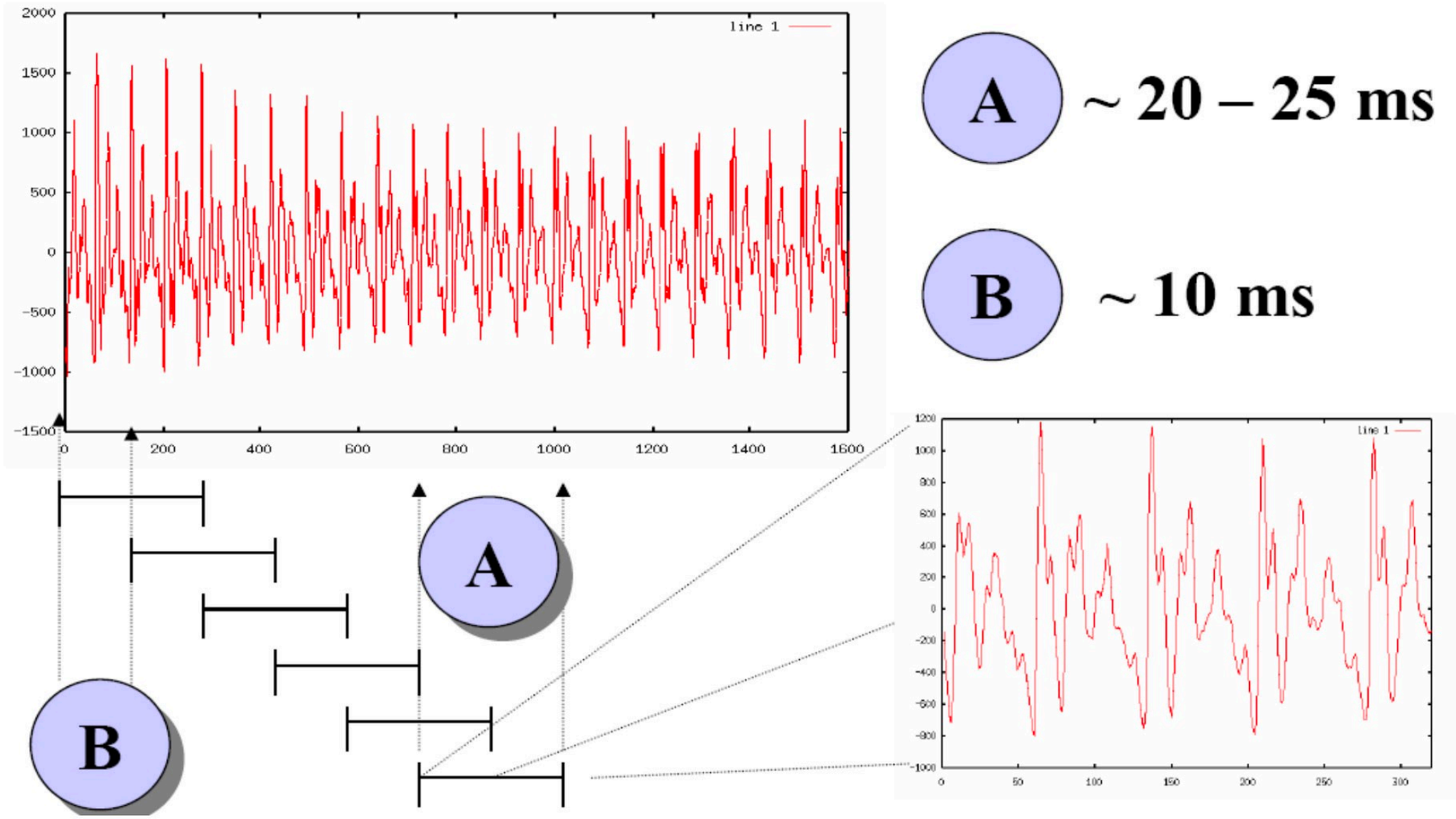
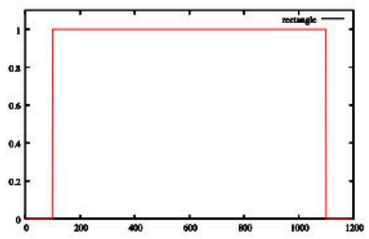
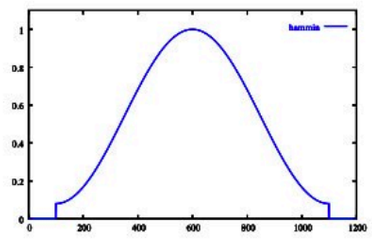


Image from Bryan Pellom

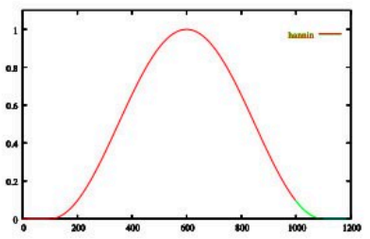
Windowing



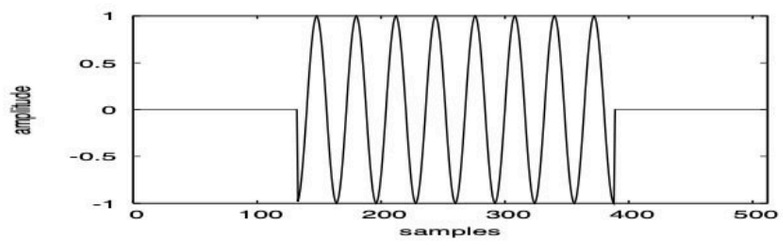
Rectangular



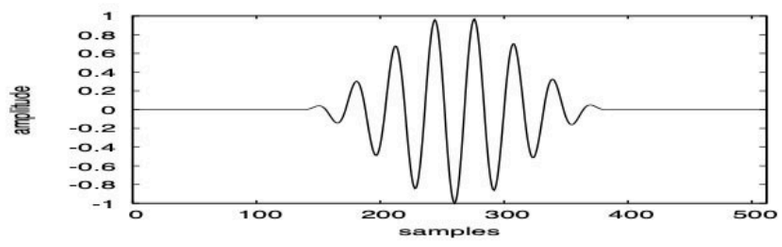
Hamming



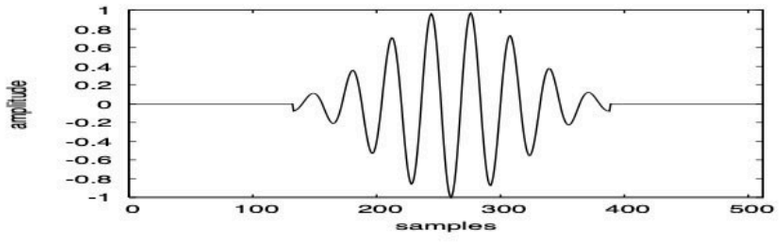
Hanning



(a) Rectangular window



(b) Hanning window



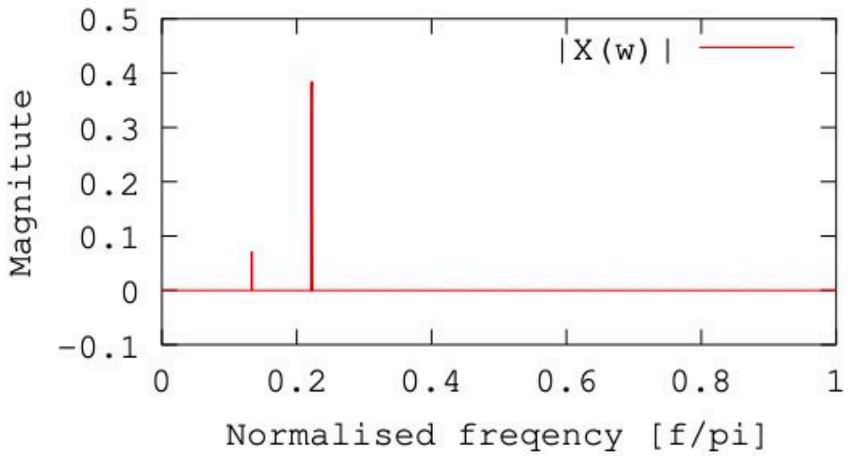
(c) Hamming window

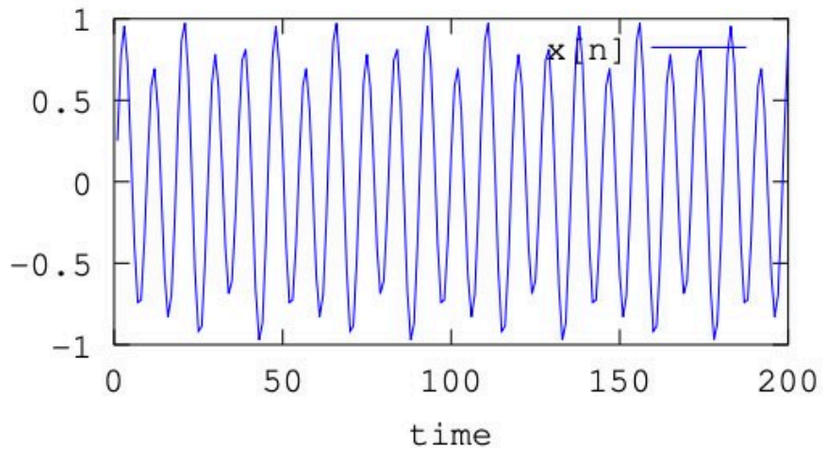
(Taylor, fig 12.1)

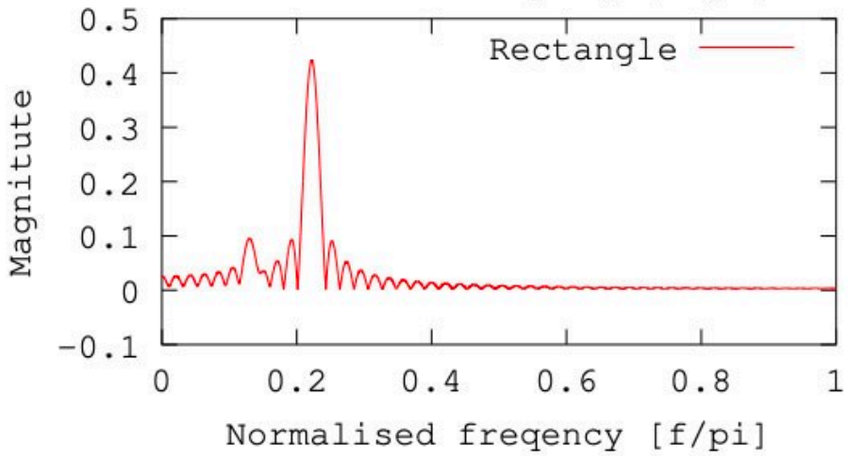
$$x[n] = w[n]\, s[n]$$

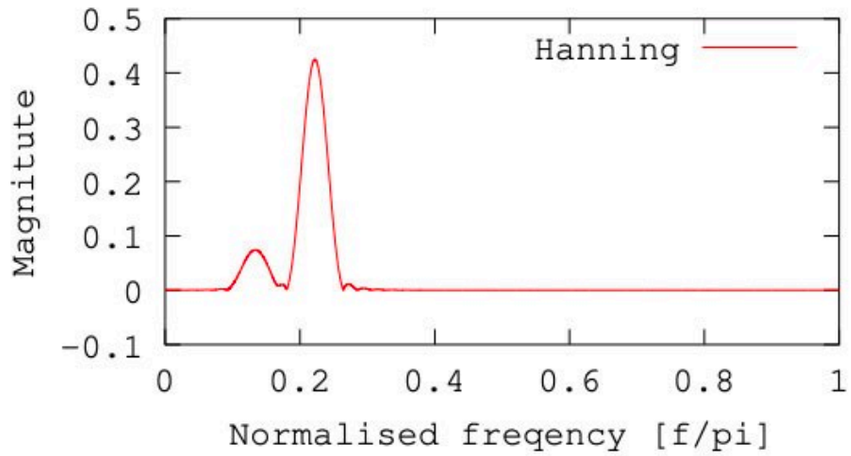
sliced frame — original audio clip

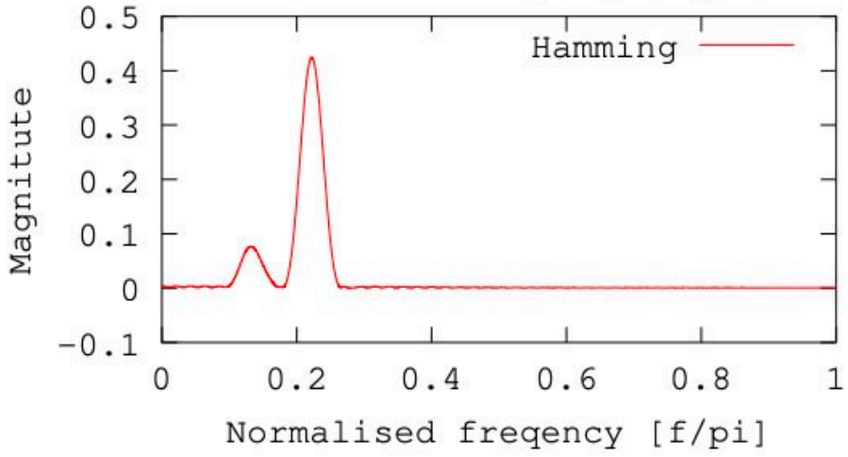
Windowing

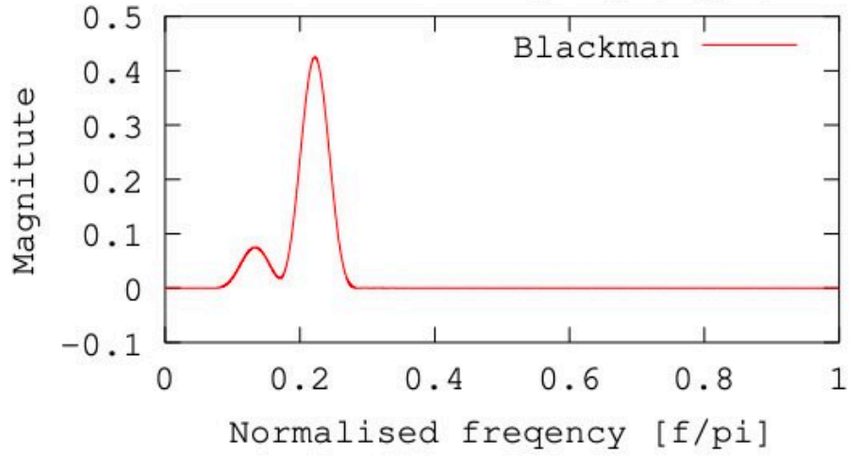

|X(w)|
Magnitute
Normalised freqency [f/pi]

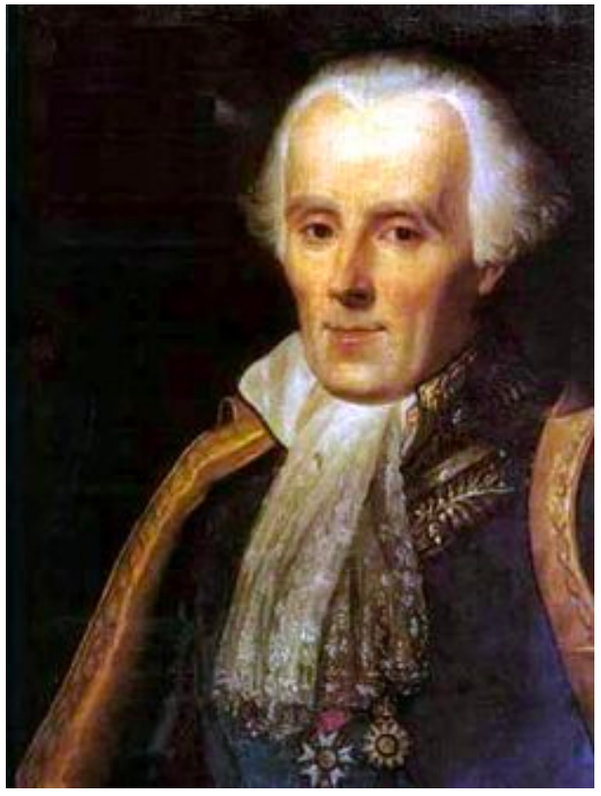

x[n]
time

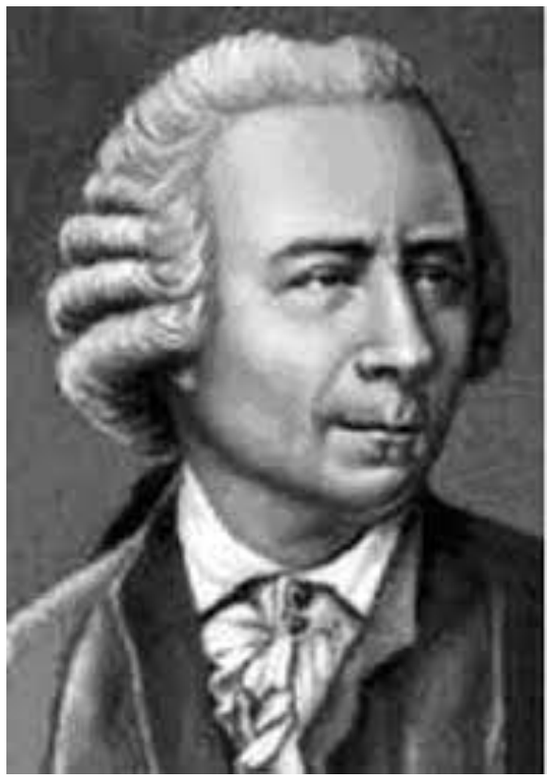

Rectangle
Magnitute
Normalised freqency [f/pi]


Hanning
Magnitute
Normalised freqency [f/pi]


Hamming
Magnitute
Normalised freqency [f/pi]


Blackman
Magnitute
Normalised freqency [f/pi]

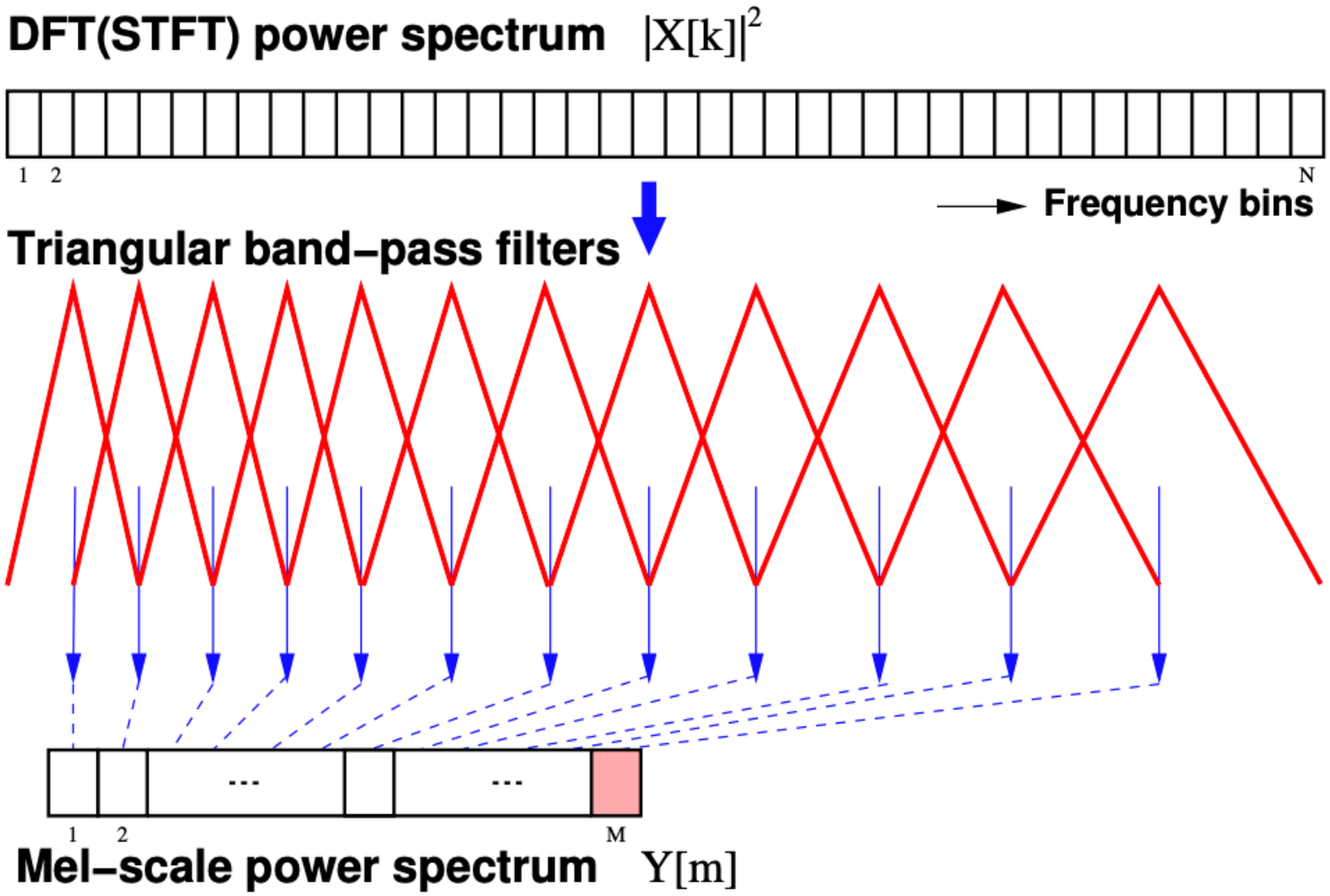




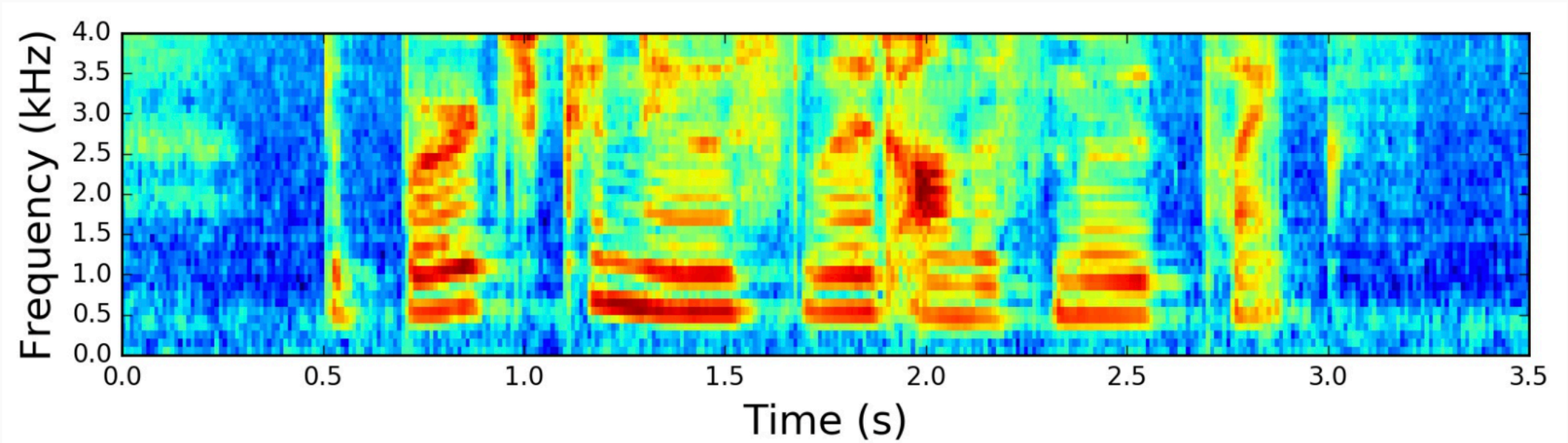
DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(-j\frac{2\pi}{N}kn\right)$$

Mel spectrogram



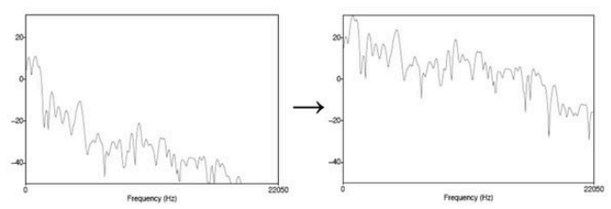
Log-Mel spectrogram

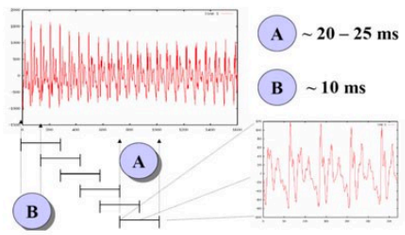


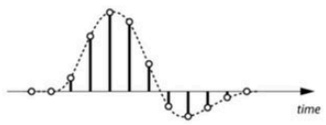
Spectrogram of the Signal

MFCC

$$y_t[n] = \sum_{m=0}^{M-1} \log(Y_t[m]) \cos\left(n(m+0.5)\frac{\pi}{M}\right)$$





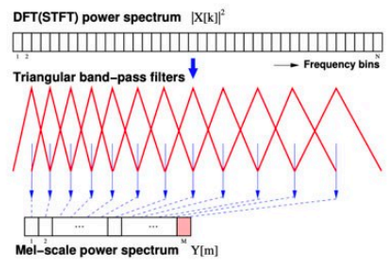


A/D conversion

Preemphasis

Windows

DFT



Mel filterbank

log( )

IDFT

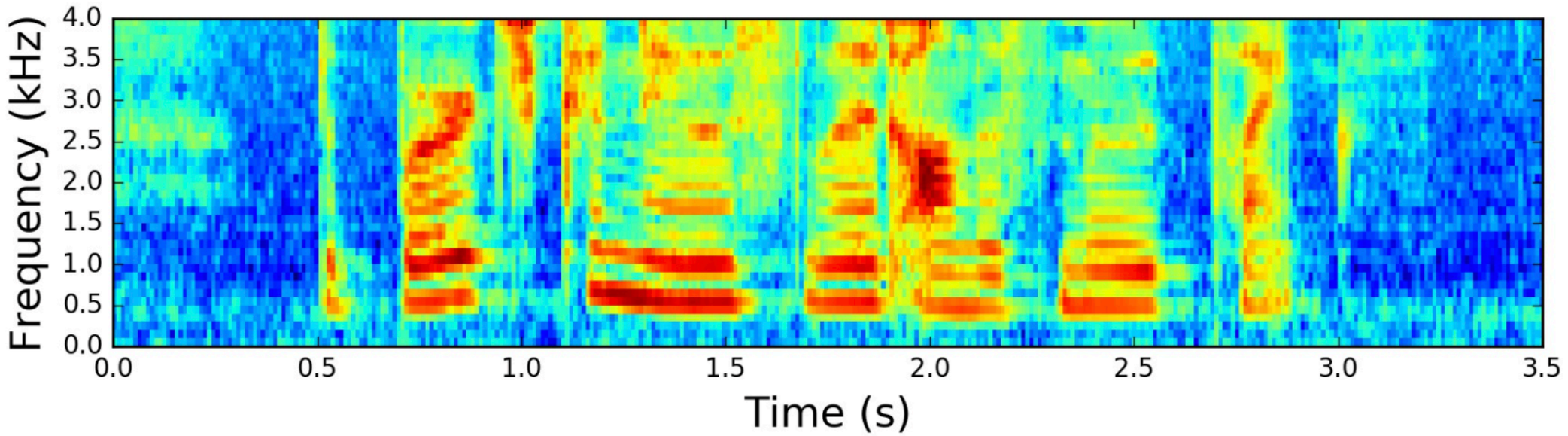
energy

Dynamic features

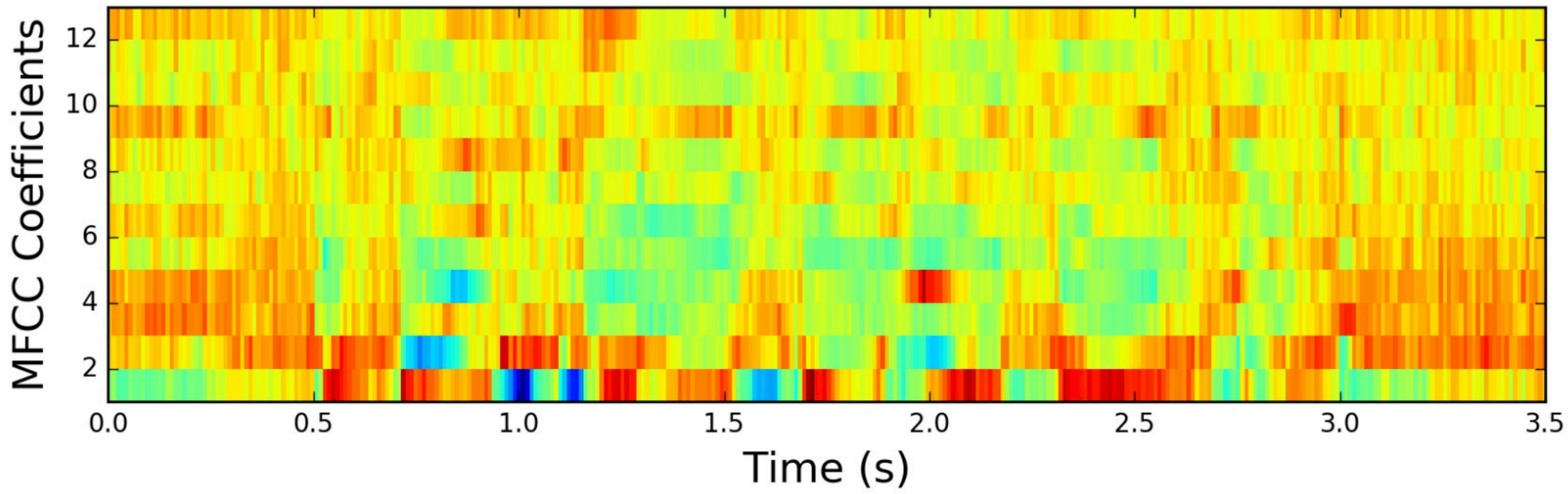
$$\begin{Bmatrix} c_t(j), e_t \\ \Delta\{c_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{c_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$$

Feature Transform

MFCC vs Log-Mel spectrogram



Spectrogram of the Signal



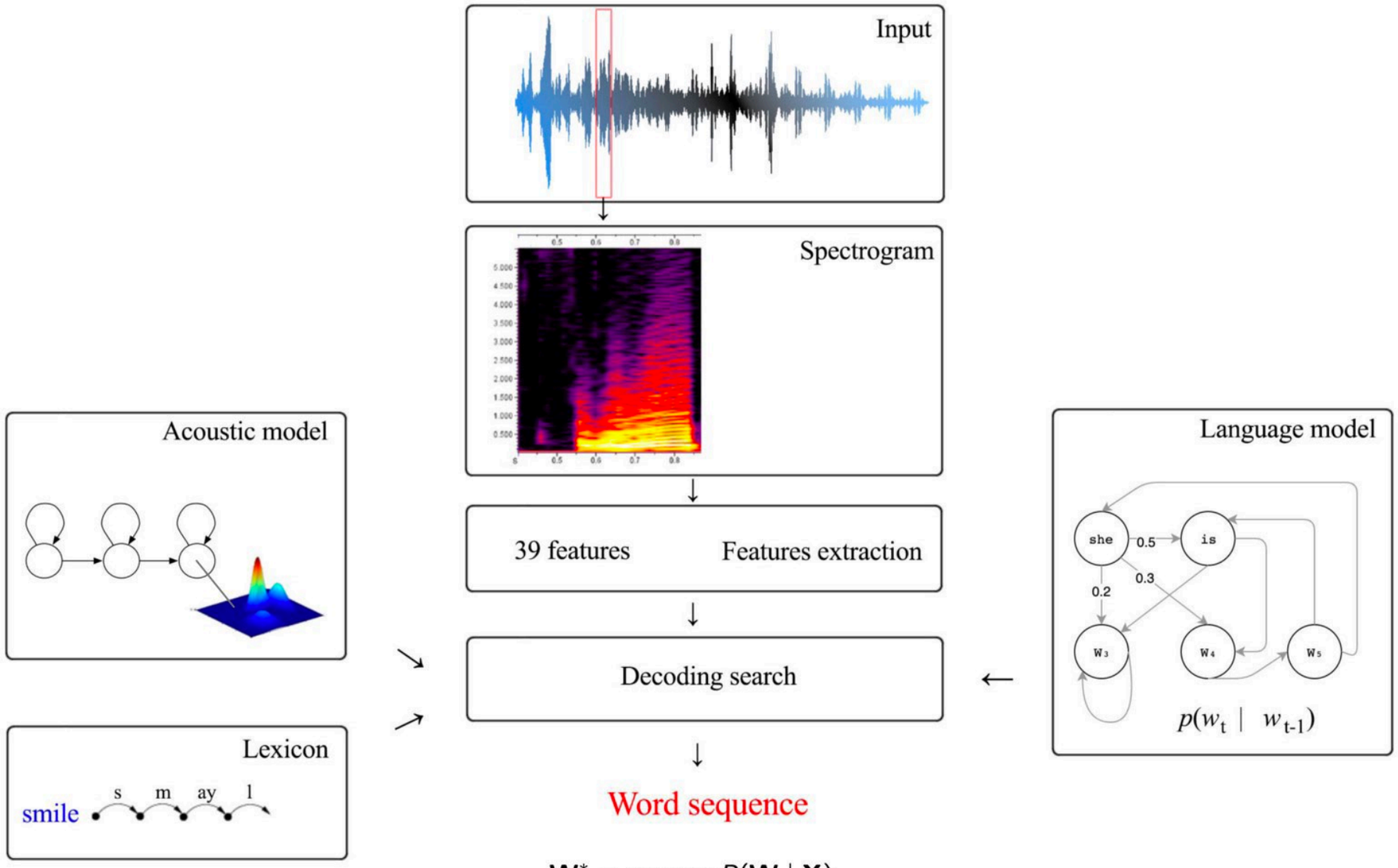
MFCCs

Audio tasks

Audio tasks

- Automatic Speech Recognition
- Emotion Recognition
- Speech Separation (Diarization)
- Speaker Recognition
- Text to Speech (TTS)
- Voice cloning
- Keyword Spotting
- Speech Enhancement

ASR

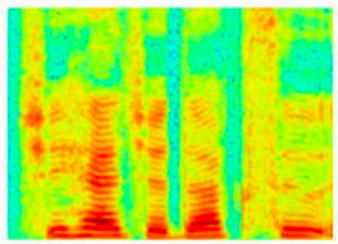


$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{X})$$

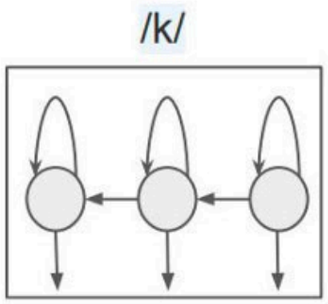
$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \; p(\mathbf{X} \mid \mathbf{W}) \;\; P(\mathbf{W})$$

word sequence acoustic model language model

ASR

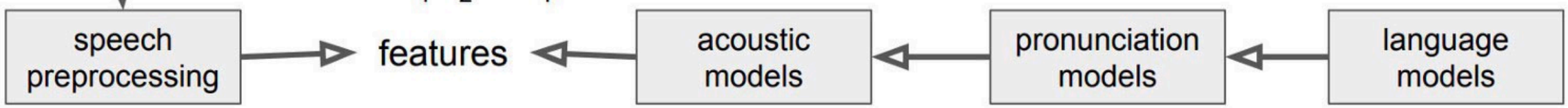


$\mathbf{X} = x_1\ x_2\ \ldots\ x_T$



Cat: /kæt/
Sits: /sɪts/
On: /ɔːn
A: /ə/
Mat: /mæt/

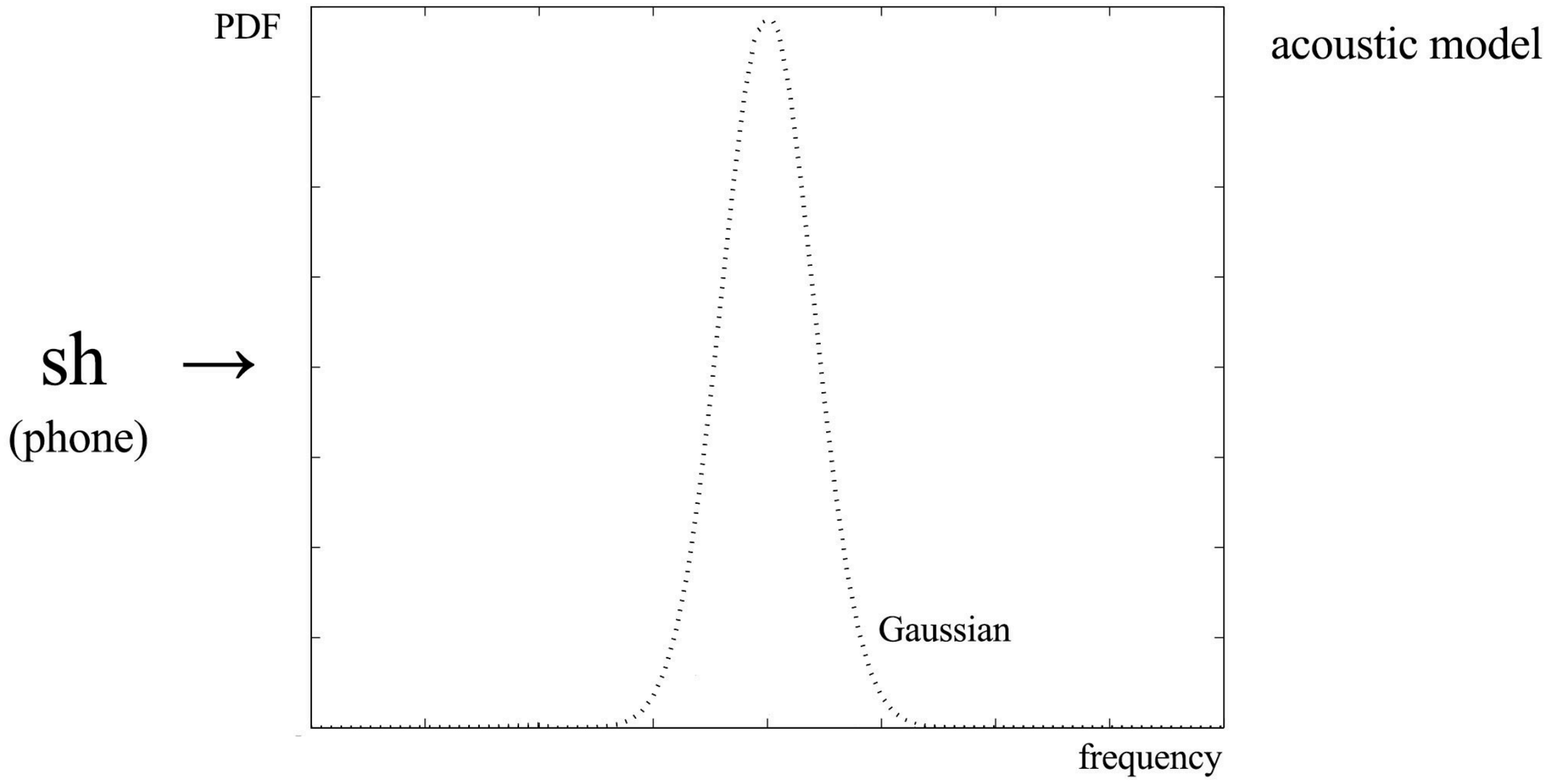
W = "*cat sits on a mat*"



$$W^* = \arg\max_W p(\mathbf{X}|\mathbf{W})\ p(\mathbf{W})$$

acoustic model — language model

GMM



GMM

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} E[(x_1-\mu_1)(x_1-\mu_1)] & E[(x_1-\mu_1)(x_2-\mu_2)] & \dots & E[(x_1-\mu_1)(x_n-\mu_n)] \\ E[(x_2-\mu_2)(x_1-\mu_1)] & E[(x_2-\mu_2)(x_2-\mu_2)] & \dots & E[(x_2-\mu_2)(x_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(x_n-\mu_n)(x_1-\mu_1)] & E[(x_n-\mu_n)(x_2-\mu_2)] & \dots & E[(x_n-\mu_n)(x_n-\mu_n)] \end{pmatrix}$$

positive semi-definite

E.g.

$$x \sim \mathcal{N}\left(\begin{pmatrix} 190 \\ 70 \end{pmatrix}, \begin{pmatrix} 100 & 25 \\ 25 & 50 \end{pmatrix}\right)$$

μ Σ

model by M Gaussian

$$b_j(\mathbf{x}) = p(\mathbf{x}|S=j) = \sum_{m=1}^{M} c_{jm}\,\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{jm},\boldsymbol{\Sigma}_{jm})$$

given phone j

weight for the Gaussian distribution m

GMM (EM-algorithm)

1. Initialize estimates for $\theta := \pi, \mu_1, \sigma_1, \mu_2, \sigma_2$

2. (**Expectation**) Compute the responsibilities for each data point

$$\gamma_i = \frac{\pi\phi(x_i; \mu_2, \sigma_2)}{(1-\pi)\phi(x_i; \mu_1, \sigma_1) + \pi\phi(x_i; \mu_2, \sigma_2)}$$

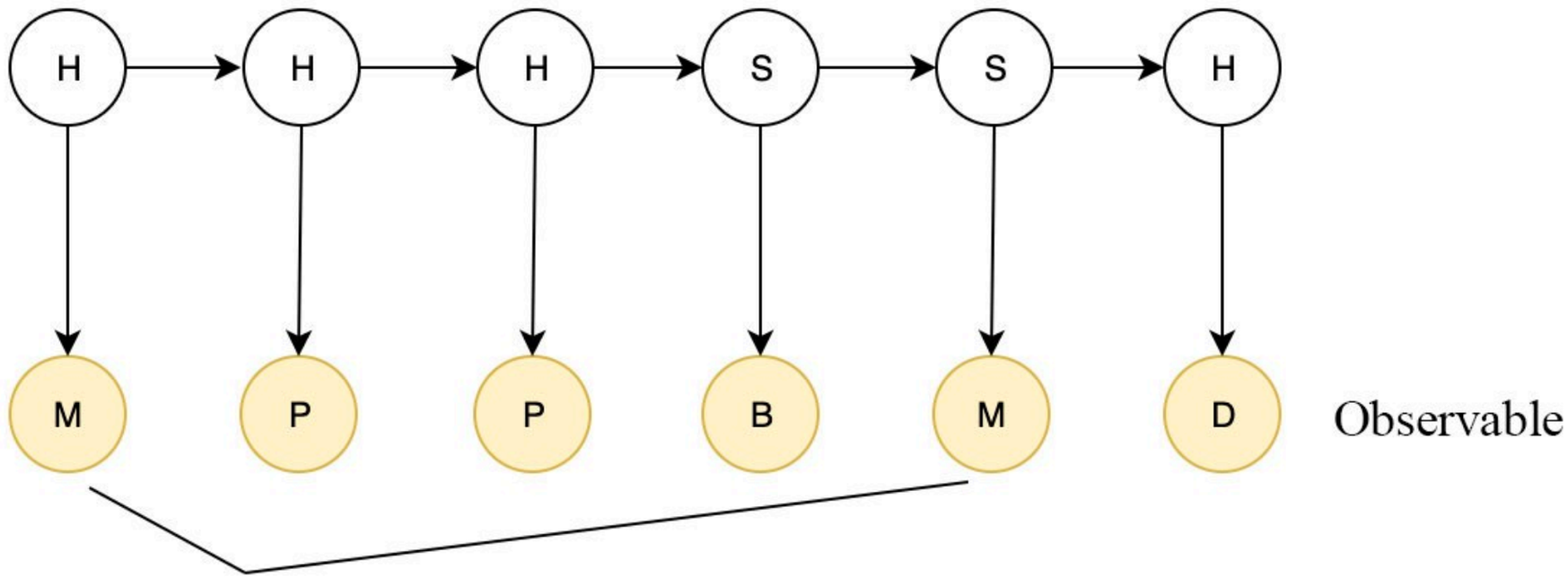
3. (**Maximization**) Update the estimates for the parameters using the maximum-likelihood estimator formula. All sums are taken across the data indexed by i and are just means/standard deviations weighted by the responsibilities γ

$$\mu_2 = \frac{\sum \gamma_i x_i}{\sum \gamma_i} \qquad \sigma_2 = \frac{\sum \gamma_i (x_i - \mu_2)^2}{\sum \gamma_i} \qquad \pi = \frac{1}{n}\sum \gamma_i$$

4. Repeat steps 2 and 3 until the parameters converge to a local optimum

HMM

internal state {H, S} is not observable or hard to determine



0.2 chance that I go to movie when I am happy.

0.4 chance that I go to movie when I am sad.

HMM

HMM models a process with a Markov process.

- It includes the initial state distribution π (the probability distribution of the initial state)
- The transition probabilities A from one state (xt) to another.
- HMM also contains the likelihood B of the observation (yt) given a hidden state. Matrix B is called the **emission probabilities**. It demonstrates the probability of our observation given a specific internal state.

HMM

π A B

$P(x_1)$

$P(x_1 = happy) = 0.8$

$P(x_1 = sad\ ) = 0.2$

x_t \ x_{t+1}

	Happy	Sad
Happy	0.99	0.01
Sad	0.1	0.9

For example, $P(Happy_{t+1}|Happy_t) = 0.99$

$P(y_t|x_t)$:

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

Observation likelihoods or Emission probabilities $\boldsymbol{B}$

Initial state distribution π

Transition probability matrix $\boldsymbol{A}$ in Markov Process

model λ

HMM

Depending on the situation, we usually ask three different types of questions regarding an HMM problem.

- Likelihood: How likely are the observations based on the current model or the probability of being at a state at a specific time step.
- Decoding: Find the internal state sequence based on the current model and observations.
- Learning. Learn the HMM model.

Likelihood: forward algorithm

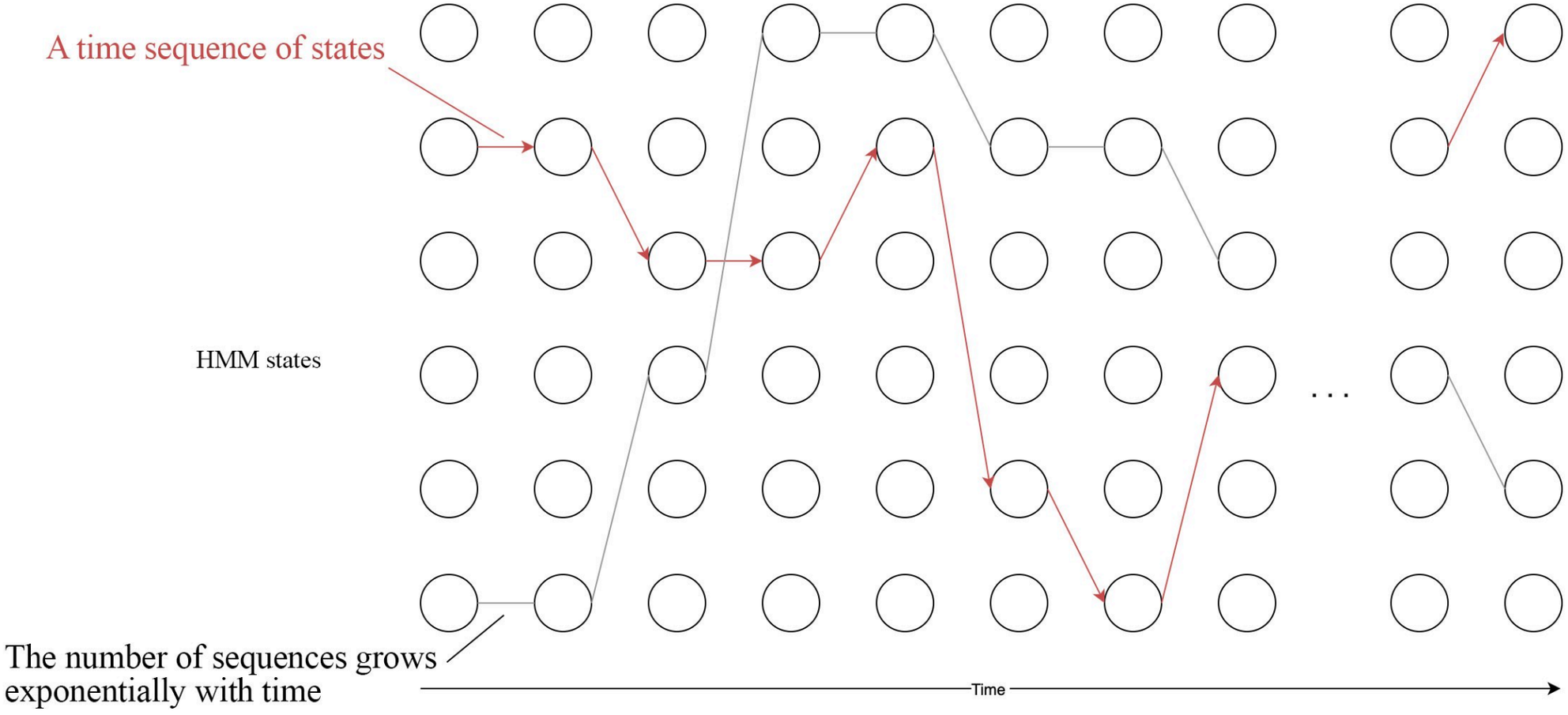
calculated from emission probability

calculated from transition probability

$$p(\mathrm{X}) = \sum_{\mathrm{S}} p(\mathrm{X},\mathrm{S}) = \sum_{\mathrm{S}} p(\mathrm{X}\,|\,\mathrm{S})\,p(\mathrm{S})$$

the observed events

sum over all possible time sequences of internal states



Likelihood: forward algorithm

HMM model parameters

$$P(x_1, x_2, \ldots, x_t \mid \lambda) = \sum_j P(x_1, x_2, \ldots, x_t, S_t = s_j \mid \lambda)$$

$\alpha_t(j)$

observations up to time t — sum over all internal states

state j — transition probability from state i to j

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\, b_j(x_t)$$

time t — sum over all internal states — emission probability

Likelihood: forward algorithm

1. initialize

initial state distribution

$$\alpha_1(j) = \pi_j b_j(x_1)$$

probability of observing y_1 given state j

2. For each time step

transition probabity

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(x_t)$$

sum over all states

probability of observing y_t given current state $= j$

probability of all previous observations give last state i

3. Result

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

all observations

final step

sum over all possible state

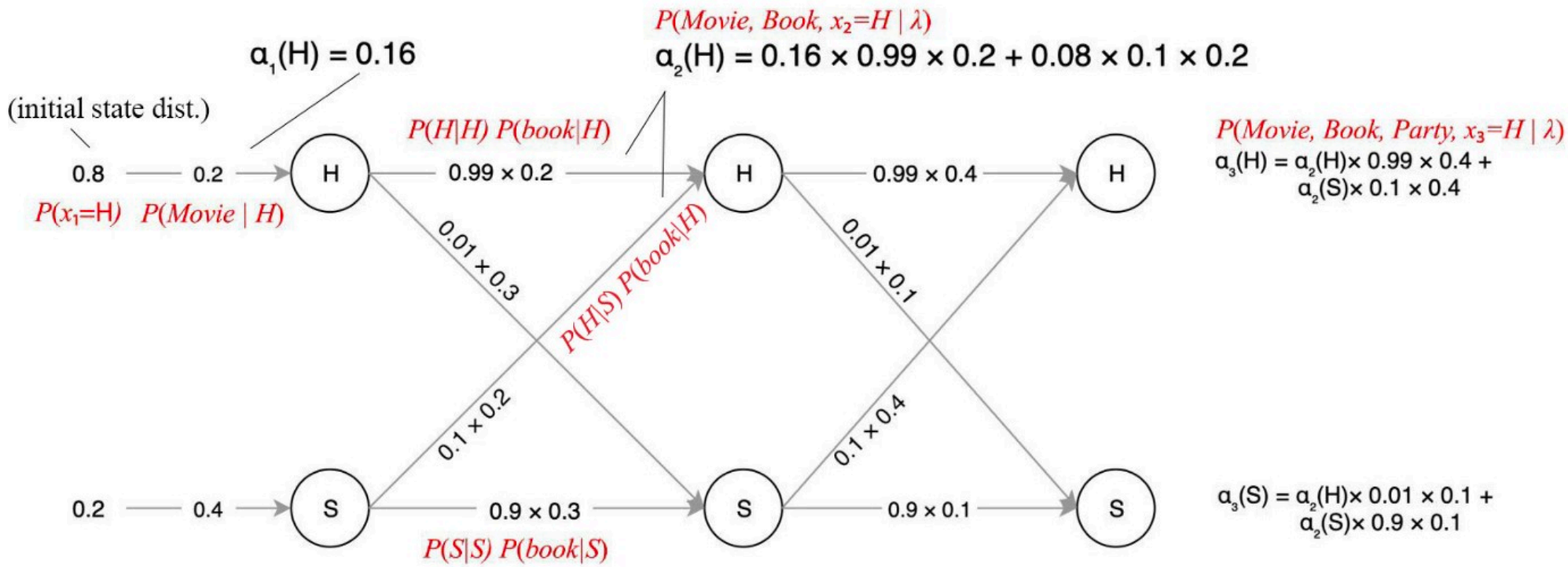
Likelihood: forward algorithm

x_{t+1}

x_t	Happy	Sad
Happy	0.99	0.01
Sad	0.1	0.9

$P(y_t|x_t)$:

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

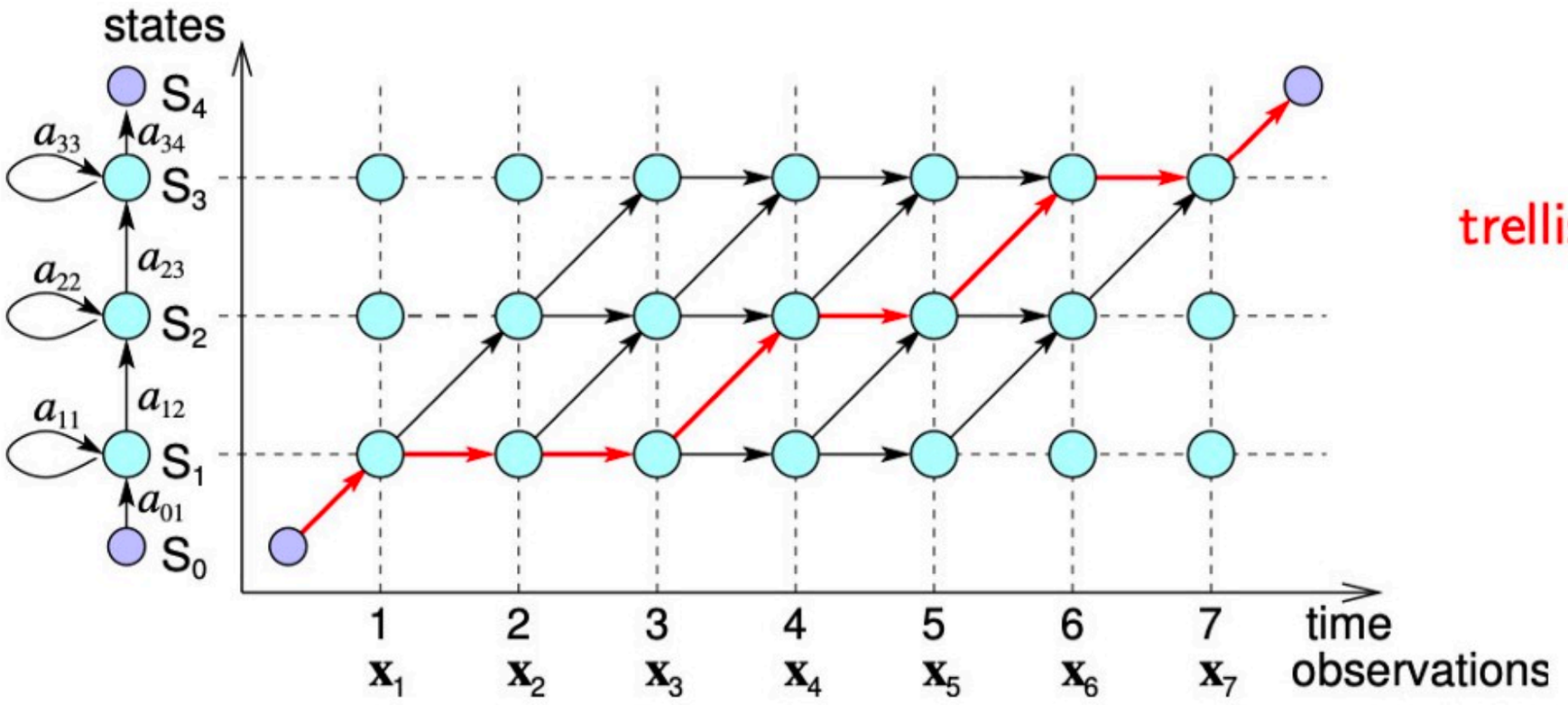


Decoding: Viterbi algorithm

$$v_t(j) = \max_{s_0, s_1 \ldots s_{t-1}} P(s_0, s_1 \ldots s_{t-1}, x_1, x_2 \ldots x_t, S_t = s_j | \lambda)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(x_t)$$

Decoding: Viterbi algorithm

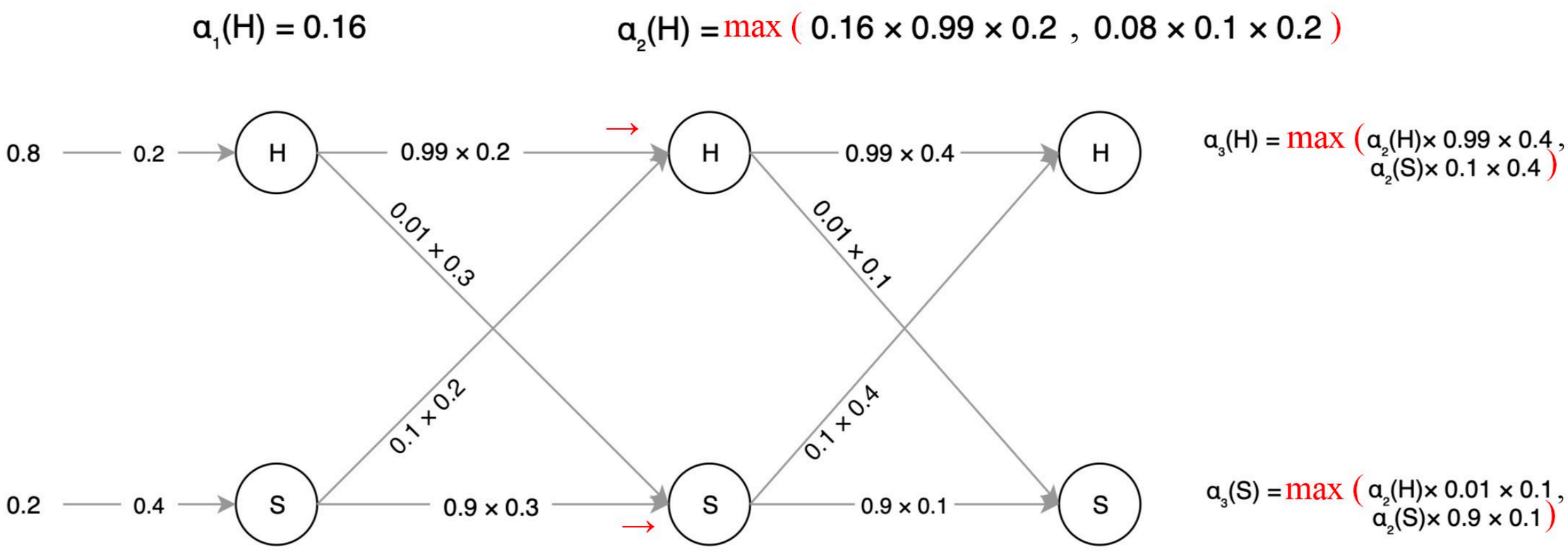


$$p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda}) = p(\mathbf{X} \,|\, \text{path}_\ell, \boldsymbol{\lambda}) P(\text{path}_\ell \,|\, \boldsymbol{\lambda})$$

likelihood: $\displaystyle\sum_{\{\text{path}_\ell\}} p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda})$

decode: $\displaystyle\max_{\text{path}_\ell} p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda})$

Decoding: Viterbi algorithm



$\alpha_1(S) = 0.08$

$\alpha_2(S) = \max (0.16 \times 0.01 \times 0.3 , 0.08 \times 0.9 \times 0.3)$

Movie

book

party

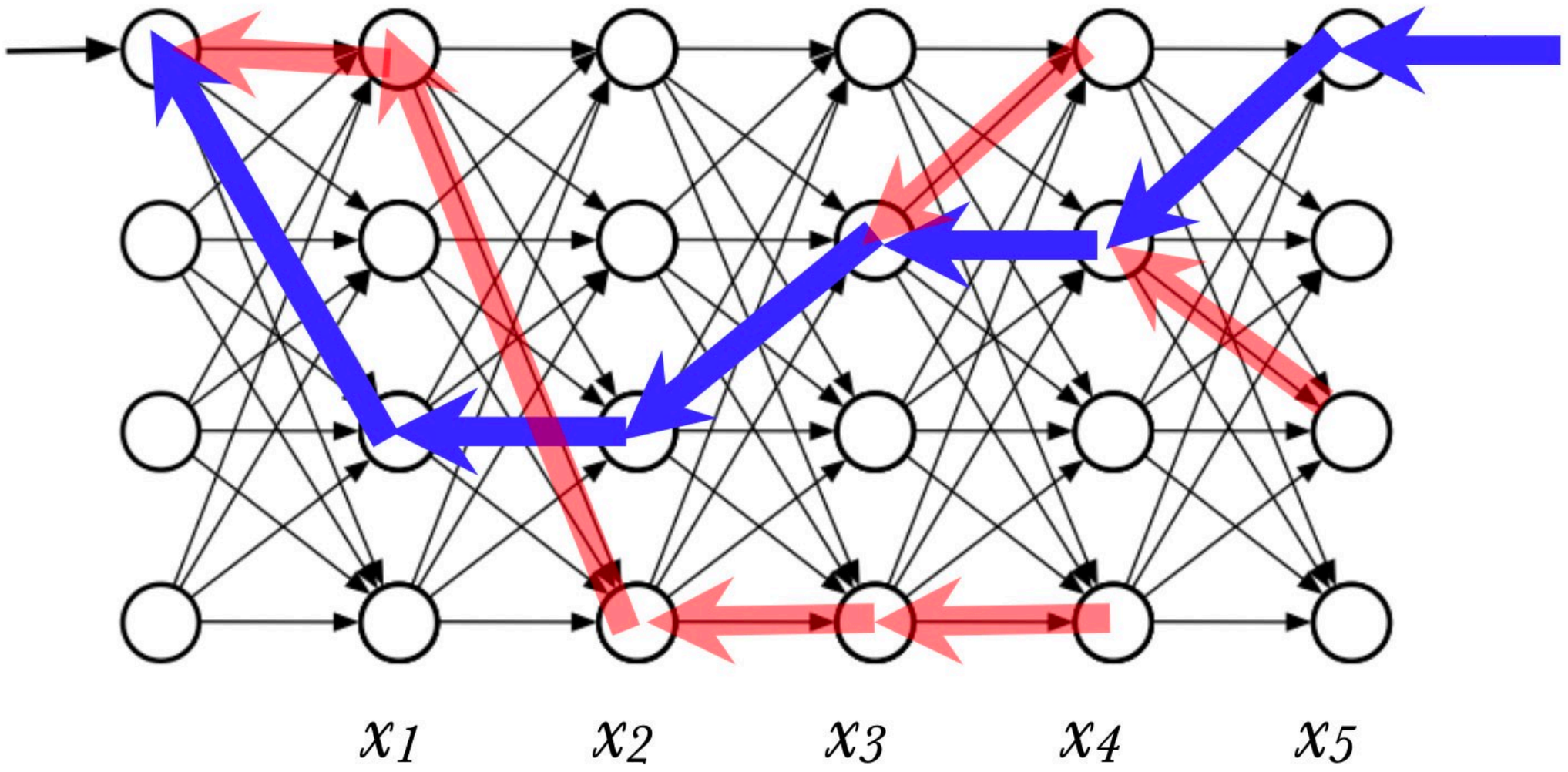
Answer = max (α_3 (H) , α_3 (S))

t=1

t=2

t=3

Decoding: Viterbi algorithm



Learning: Baum–Welch algorithm

$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta)$ (Given at time t with internal state i, the probability of seeing all the observations so far.)

$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|X_t = i, \theta)$ (Given at time t with internal state i, the probability of seeing all the coming observations.)

$\gamma_i(t) = P(X_t = i|Y, \theta)$ (Probability at a given state i at time t given the observations)

$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta)$ (Probability of trainsition from state i to state j at time t given the observations)

α - forward probability

β - backward probability

Learning: backward algorithm

$$\beta_i(T) = 1$$

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij}\, b_j(y_{t+1})\, \beta_j\,(t+1) \qquad \text{(For each time step from the end)}$$

$$P(Y|\lambda) = \sum_{j=1}^{N} \pi_j\, b_j(y_1)\, \beta_j(1)$$

Learning: Baum–Welch algorithm

Given the HMM model parameters fixed, we can apply the forward and backward algorithm to calculate α and β from the observations. γ can be calculated by simply multiplying α with β, and then renormalize it.

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{P(X_t = i, Y|\theta)}{P(Y|\theta)}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

Learning: Baum–Welch algorithm

ξ is the probability of transiting from state i to j after time t given all the observations. It can be computed by α and β similarly.

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$$

$$= \frac{P(X_t = i, X_{t+1} = j, Y|\theta)}{P(Y|\theta)}$$

$$= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}$$

Learning: Baum–Welch algorithm

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^{T} \mathbf{1}_{y_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

Learning: Baum–Welch algorithm

Estimation step

Forward

1. $\alpha_i(1) = \pi_i b_i(y_1)$,
2. $\alpha_i(t+1) = b_i(y_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji}$.

Backward

1. $\beta_i(T) = 1$,
2. $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(y_{t+1})$.

Update

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{P(X_t = i, Y|\theta)}{P(Y|\theta)}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

equals α (forward) × β (backward) for state *i* at time *t*

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta)$$

$$= \frac{P(X_t = i, X_{t+1} = j, Y|\theta)}{P(Y|\theta)}$$

$$= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}$$

equals α for state *i* at time *t* × tranistion prob. between *i* and *j* × β for state *j* at time *t+1* × observe y_{t+1} for state *j*

Maximization step

$$\pi_i^* = \gamma_i(1)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$P(X_t = i, X_{t+1} = j|Y, \theta)$

$P(X_t = i|Y, \theta)$

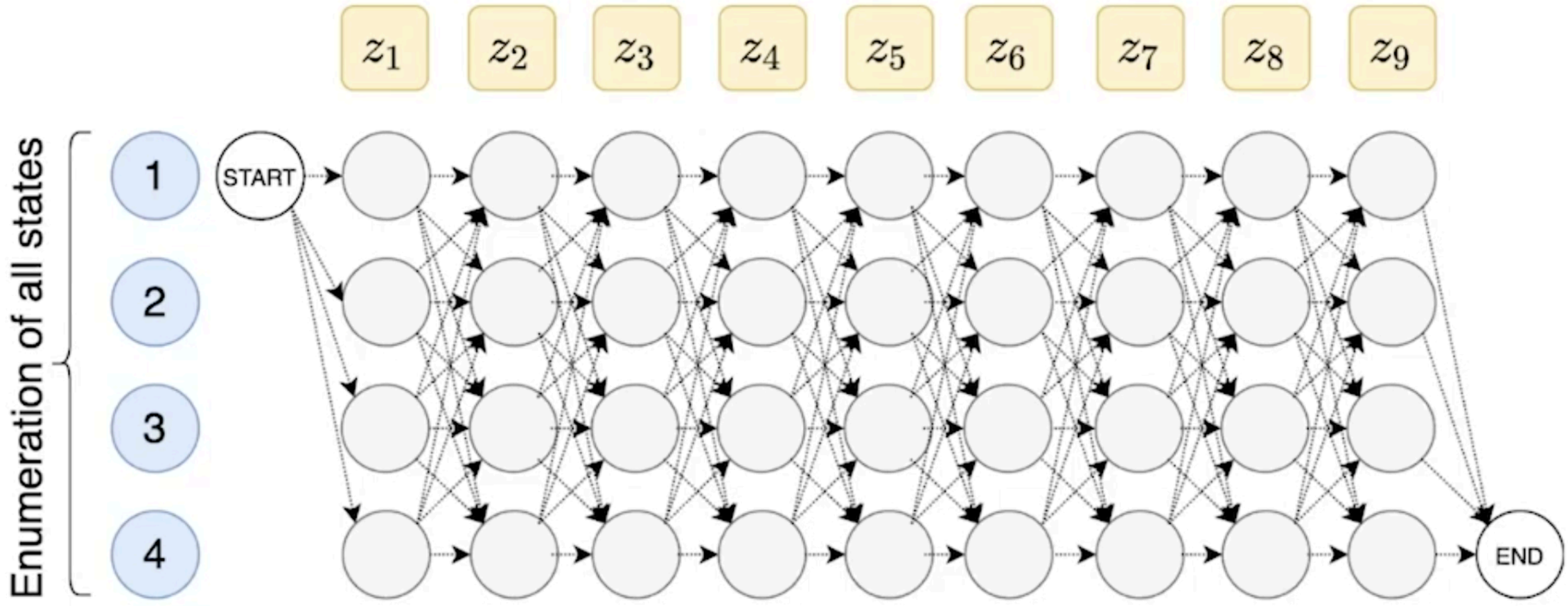
$$b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

(sum γ over all time steps where the observation y_t is the same as v_t at time *t*)

where

$$1_{y_t = v_k} = \begin{cases} 1 & \text{if } y_t = v_k, \\ 0 & \text{otherwise} \end{cases}$$

Inference trellis



Training trellis