Understanding GMM, HMM, DNN and LSTM

Pradeep R

12th April 2019

Outline

- Introduction
- K-Means Clustering
- Gaussian Model
- Need for GMM
- Need for HMM
  - Implementation
- Conclusion

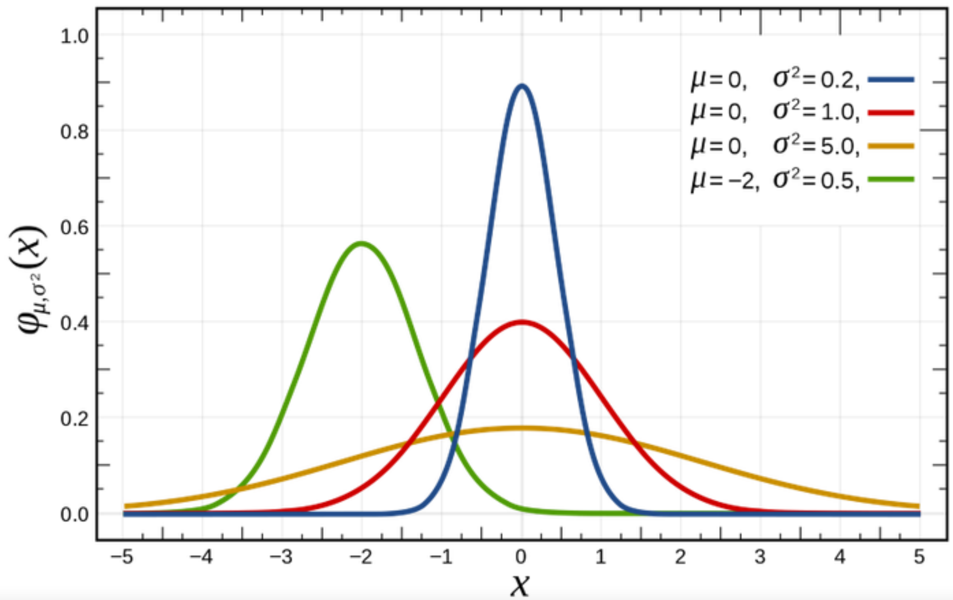
Introduction

- Applications:
  - Hand Character Recognition
  - Spoken Digit Recognition
  - Weather Prediction
  - Human activity detection
- Central limit theorem states that “when we add large number of independent random variables, irrespective of the original distribution of these variables, their normalized sum tends towards a Gaussian distribution.”
- For example, the distribution of total distance covered in an random walk tends towards a Gaussian probability distribution.

Why Gaussian?

- Gaussian remain as a Gaussian even after transformation
  - Product of two Gaussian is a Gaussian
  - Sum of two independent Gaussian random variables is a Gaussian
  - Convolution of Gaussian with another Gaussian is a Gaussian
  - Fourier transform of Gaussian is a Gaussian
- Simpler
  - The entire distribution can be specified using just two parameters- mean and variance

Variants of Gaussian



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mu - x)^2}{2\sigma^2}}$$

UniVariate Gaussian

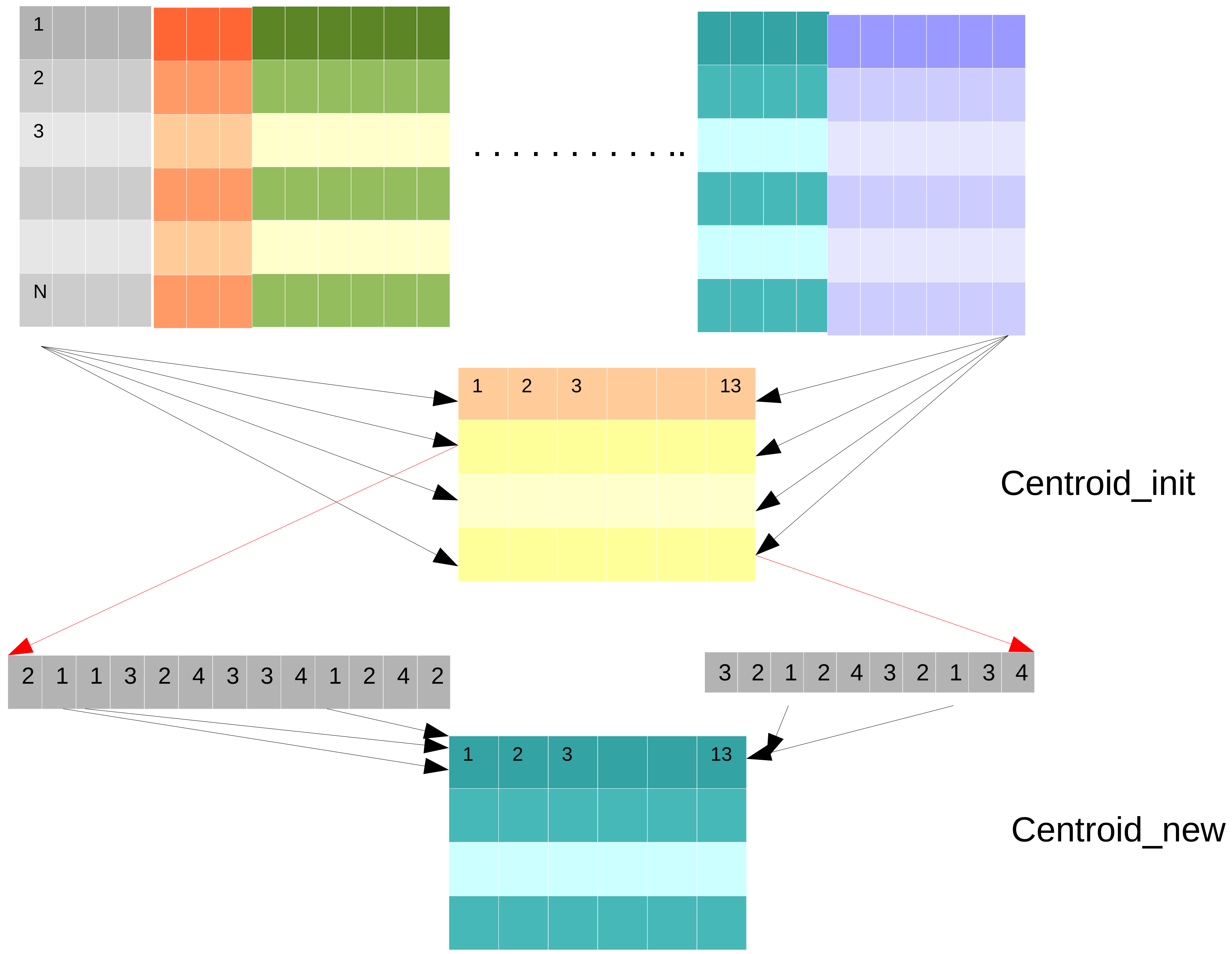
$$p(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

MultiVariate Gaussian

$$f(\mathbf{x}) = \sum_{i=1}^{N} p_i \cdot \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma_i)}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{m}_i)\, \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i)^T \right)$$

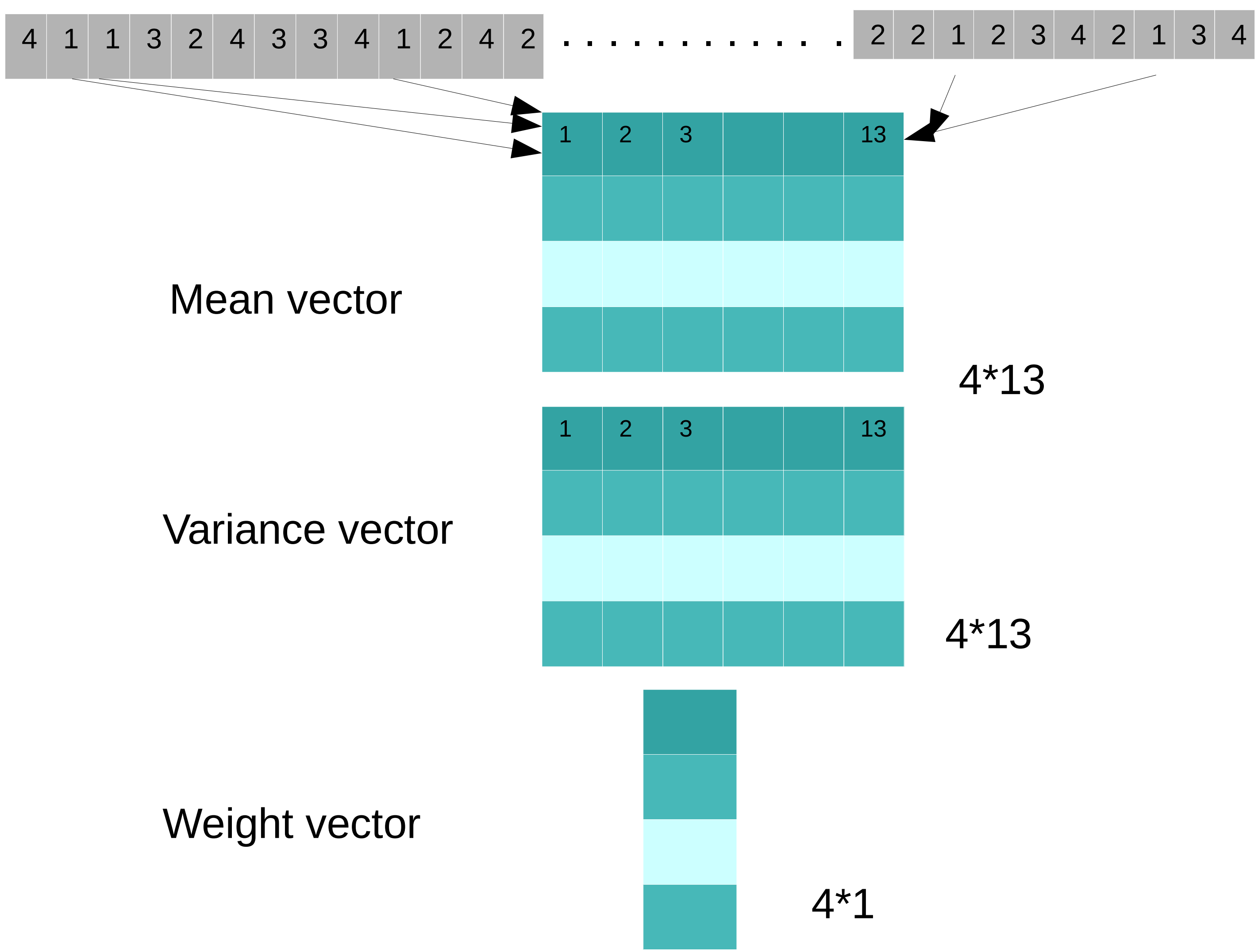
Gaussian Mixture Model

K-Means Clustering - Idea



Error
10e5
10e4
0

Error=Sum(Sum(centroid_new-centroid_old).^2)



GMM Training: Expectation Maximization Algorithm

- Initialization: k-means clustering
- E-step

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- M-step

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \qquad \pi_k^{\text{new}} = \frac{N_k}{N}$$

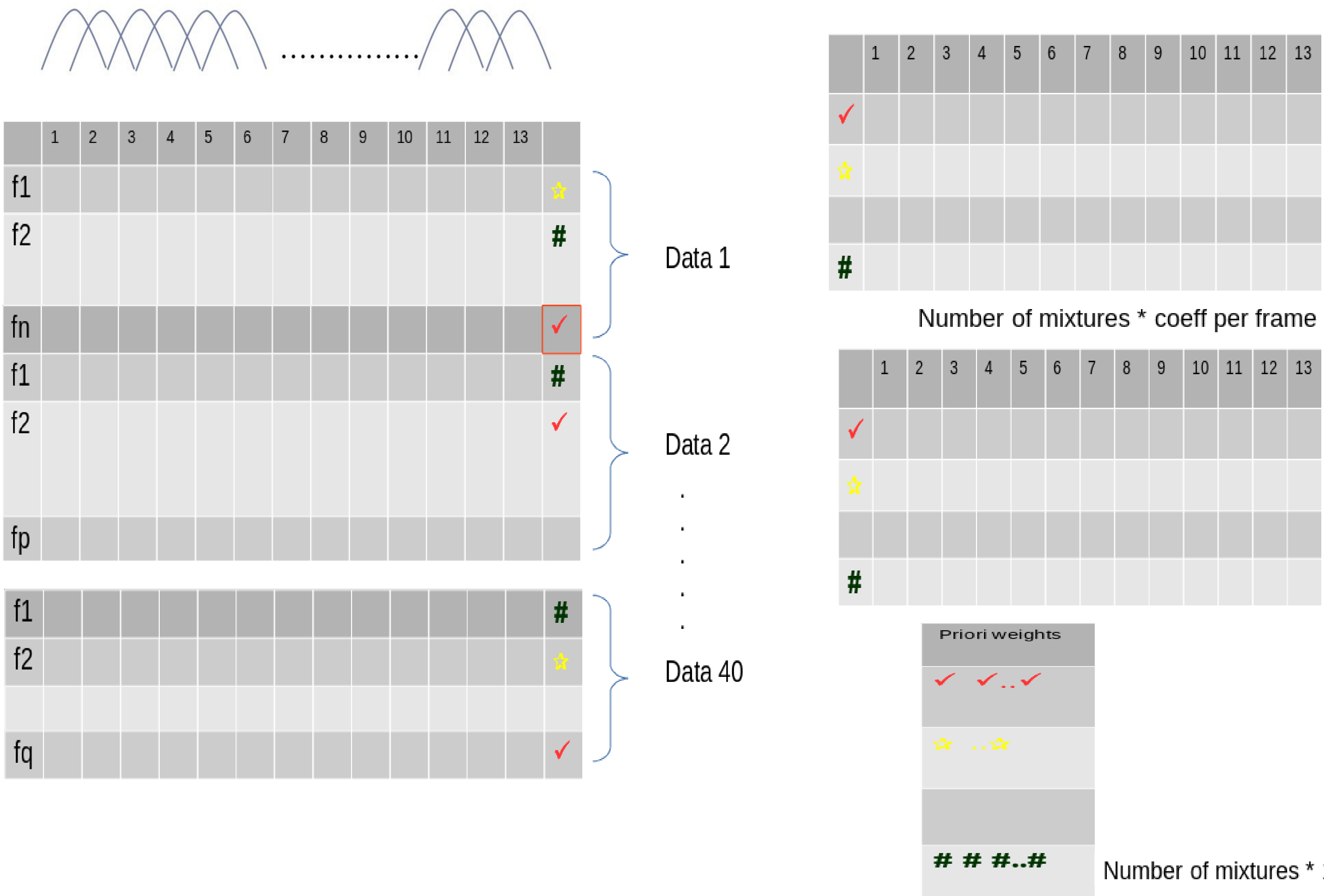
$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

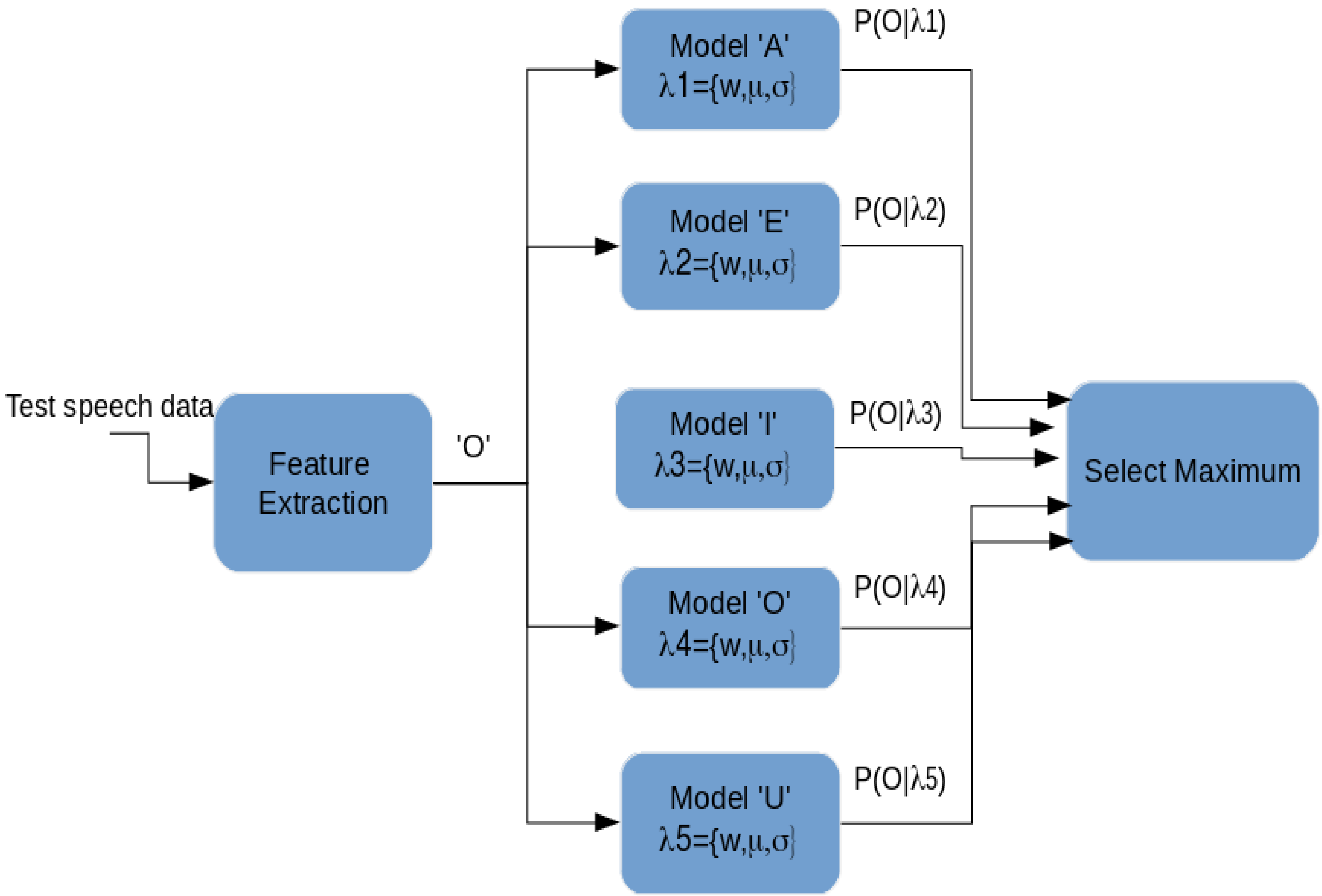
- Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Implementation-Initial Setup



Testing



Test speech data
Feature Extraction
'O'
Model 'A' λ1={w,μ,σ}
Model 'E' λ2={w,μ,σ}
Model 'I' λ3={w,μ,σ}
Model 'O' λ4={w,μ,σ}
Model 'U' λ5={w,μ,σ}
P(O|λ1)
P(O|λ2)
P(O|λ3)
P(O|λ4)
P(O|λ5)
Select Maximum

Drawbacks of GMM

- Cannot alone perform time series prediction
- It is necessary to evaluate probabilities along with time in many practical applications.
  - Ex 01: GMM build for word ‘Kamal’ and ‘Kalam’ gathers no sequential information
  - GMM - ‘b’ and GMM - ‘d’ may have false substitutions
- Hence it is necessary to capture temporal/sequential information along with spatial information.

Hidden Markov Models

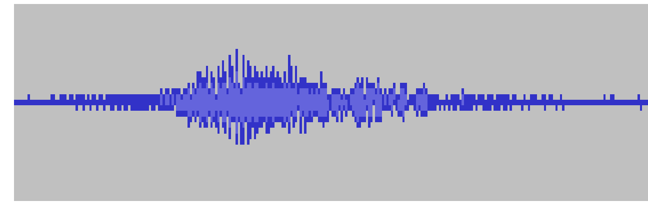
- Terminologies:
  - State
  - State Transition
  - Emission Probability

$$b_j(\boldsymbol{o}_t) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_j)}$$

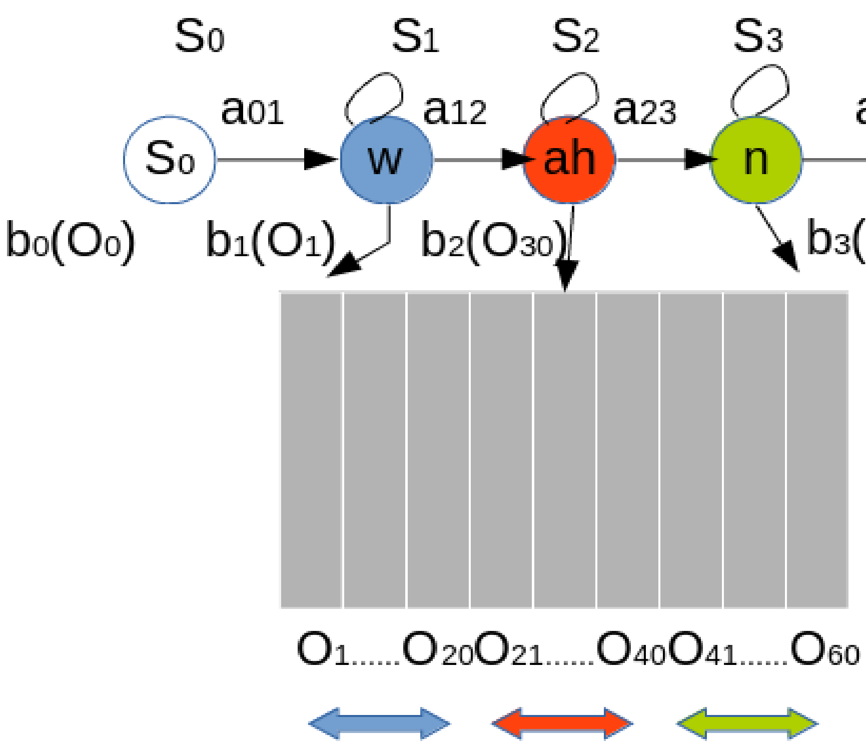
$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{o}_t$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{1}{T} \sum_{t=1}^{T} (\boldsymbol{o}_t - \boldsymbol{\mu}_j)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)'$$

Illustration of phone level modeling



- Train utterance.wav
  - Trancription: One
  - Phonetic equivalence: One -- w ah n



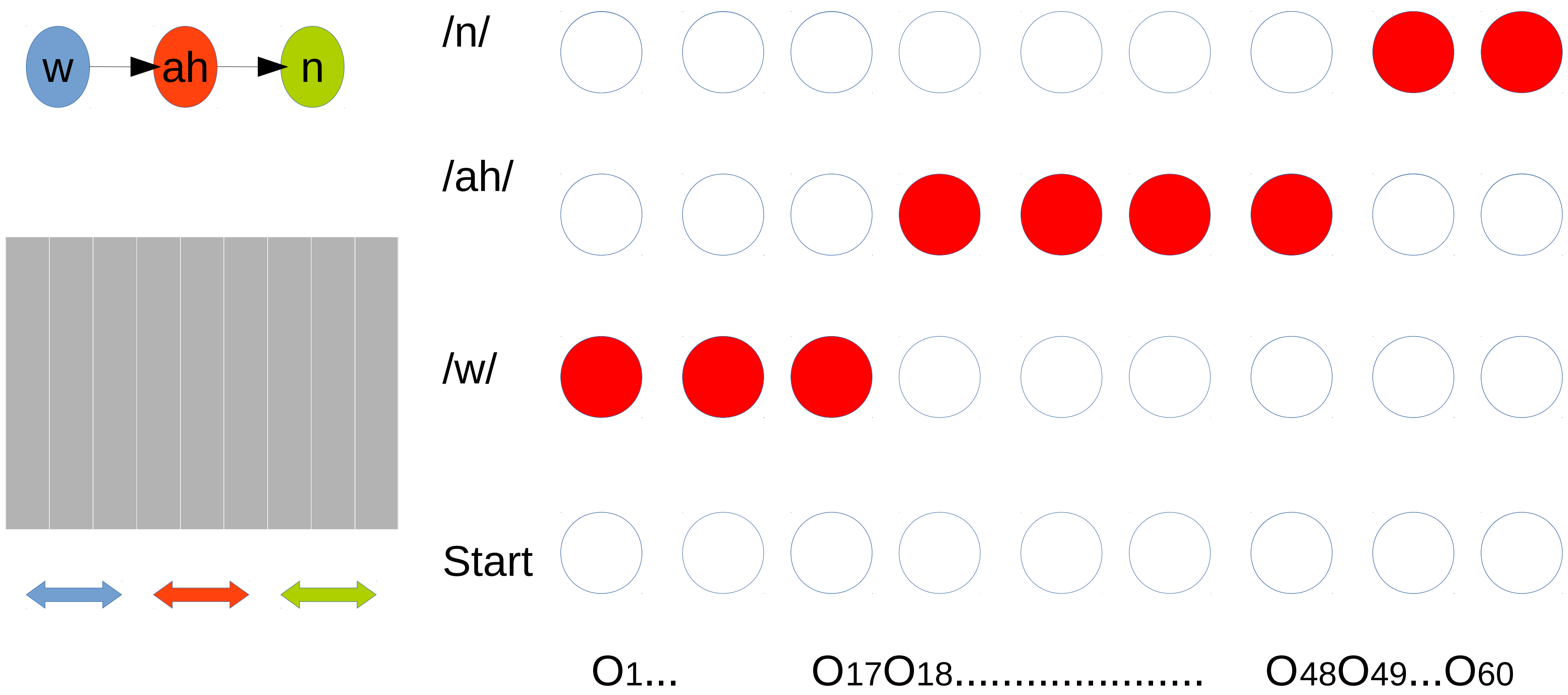
$b_j(O_0)=1$

$$b_j(\boldsymbol{o}_t) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\boldsymbol{o}_t-\boldsymbol{\mu}_j)'\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_j)}$$

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{o}_t$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{o}_t - \boldsymbol{\mu}_j)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)'$$

Expected Outcome



A_{ij} =

	S_0	S_1	S_2	S_3
S_0	1	0	0	0
S_1				
S_2				
S_3				

B_{jk} =

$O_0 O_1 ...... O_{20} O_{21} ...... O_{40} O_{41} ...... O_{60}$

S_0	1	0	0	..	0	..	..	0	0
S_1	0								
S_2	0								
S_3	0								

Compute Forward probabilities: Forward Recursion

- The goal here is to find the probablity that the HMM is in state '*i*' at time *'t'* ----- *$\alpha_i(t)$*

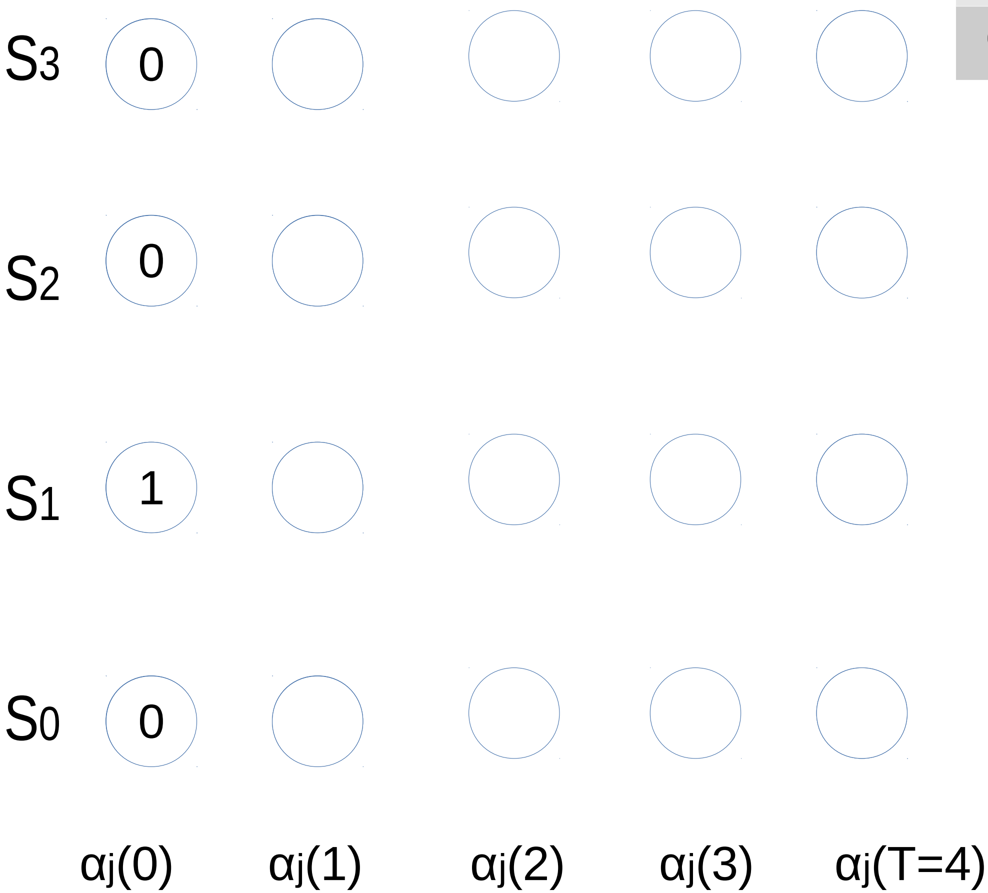
$$\alpha_j(t) = \begin{cases} 0 \;;\; t=0 \;\&\; j!=\text{initial state} \\ 1 \;;\; t=0 \;\&\; j=\text{initial state} \\ \text{sum}(\alpha_i(t-1)*a_{ij})*b_j(O_t) \end{cases}$$

$$\beta_i(t) = \begin{cases} 0 \;;\; S_i(t)!=0 \;\&\; t=\text{final state} \\ 1 \;;\; S_i(t)=0 \;\&\; t=\text{final state} \\ \text{sum}(\alpha_j(t+1)*a_{ij})*b_j(O_{t+1}) \end{cases}$$

$$\alpha_j(t) = \begin{cases} 0 \ ; \ t=0 \ \& \ j \neq \text{initial state} \\ 1 \ ; \ t=0 \ \& \ j = \text{initial state} \\ \text{sum}(\alpha_i(t-1) * a_{ij}) * b_j(O_t) \end{cases}$$

- $O^4 = \{O_1, O_{25}, O_{46}, O_0\}$

  t: 1 2 3 4



S_0	S_1	S_2	S_3
1	0	0	0
0.2	0.3	0.1	0.4
0.2	0.5	0.2	0.1
0.7	0.1	0.1	0.1

[A]

O_0	O_1	O_{25}	O_{46}	O_0
1	0	0	0	0
0	0.3	0.1	0.4	0.2
0	0.1	0.7	0.1	0.1
0	0.5	0.1	0.2	0.2

[B]

<u>Assumption:</u>

- Let the HMM be at state S1 at t=0
- $\alpha_1(0)=1$
- Find $P(O^4|\theta)$

S_3	0	0	0	0	0.0018
S_2	0	0.09	0.0052	0.0052	0
S_1	1	0.01	0.0217	0.0005	0
S_0	0	0.2	0.0057	0.001	0

Use **Viterbi decoding** to find the hidden state sequence that generated the particular observation

$P(O^4|\theta)=\alpha 0(T)=0.0018$

$P(O^4)=\beta i(0)$

t=1

$\alpha 0(1)=\{\alpha 0(0)a00+\alpha 1(0)a10+\alpha 2(0)a20+\alpha 3(0)a30\}*b0(O1)=0$

$\alpha 1(1)=\{\alpha 0(0)a01+\alpha 1(0)a11+\alpha 2(0)a21+\alpha 3(0)a31\}*b1(O25)=0.09$

$\alpha 2(1)=\{\alpha 0(0)a02+\alpha 1(0)a12+\alpha 2(0)a22+\alpha 3(0)a32\}*b2(O46)=0.02$

$\alpha 3(1)=\{\alpha 0(0)a03+\alpha 1(0)a13+\alpha 2(0)a23+\alpha 3(0)a33\}*b3(O0)=0.05$

In matrix form [0 1 0 0]*[A]*[B(:,2)] = [0.2 0.01 0.09 0]

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\gamma_i(t) = P(X_t = i | Y, \theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

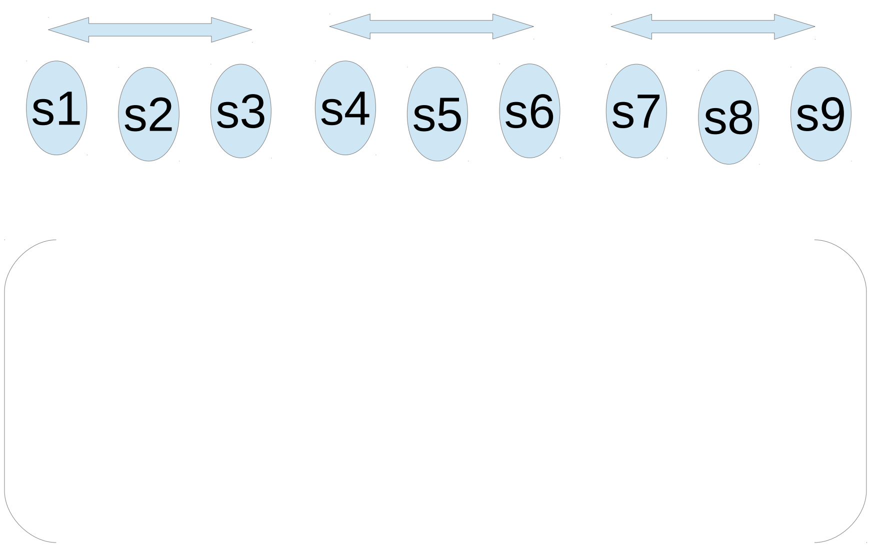
$$\hat{b}_j(o_k) = \frac{\sum_{t:O_t=o_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

ξt(i,j) =

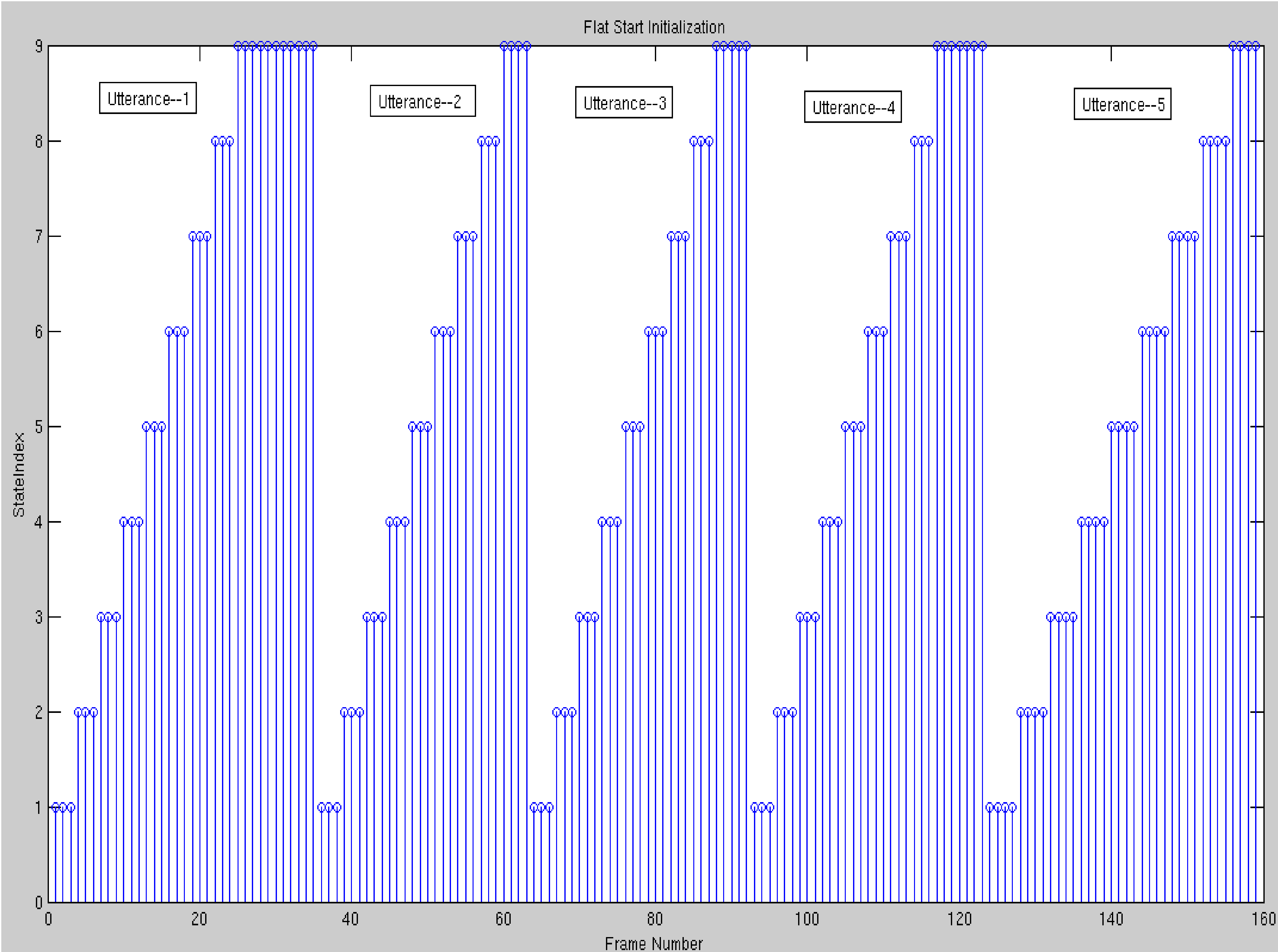
$$\frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k(t)a_{kl}\beta_l(t+1)b_l(y_{t+1})} = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}$$

Illustration

- Data:
  - Ten words of the utterance one
  - One : w ah n
- Build 3 state HMM per phone

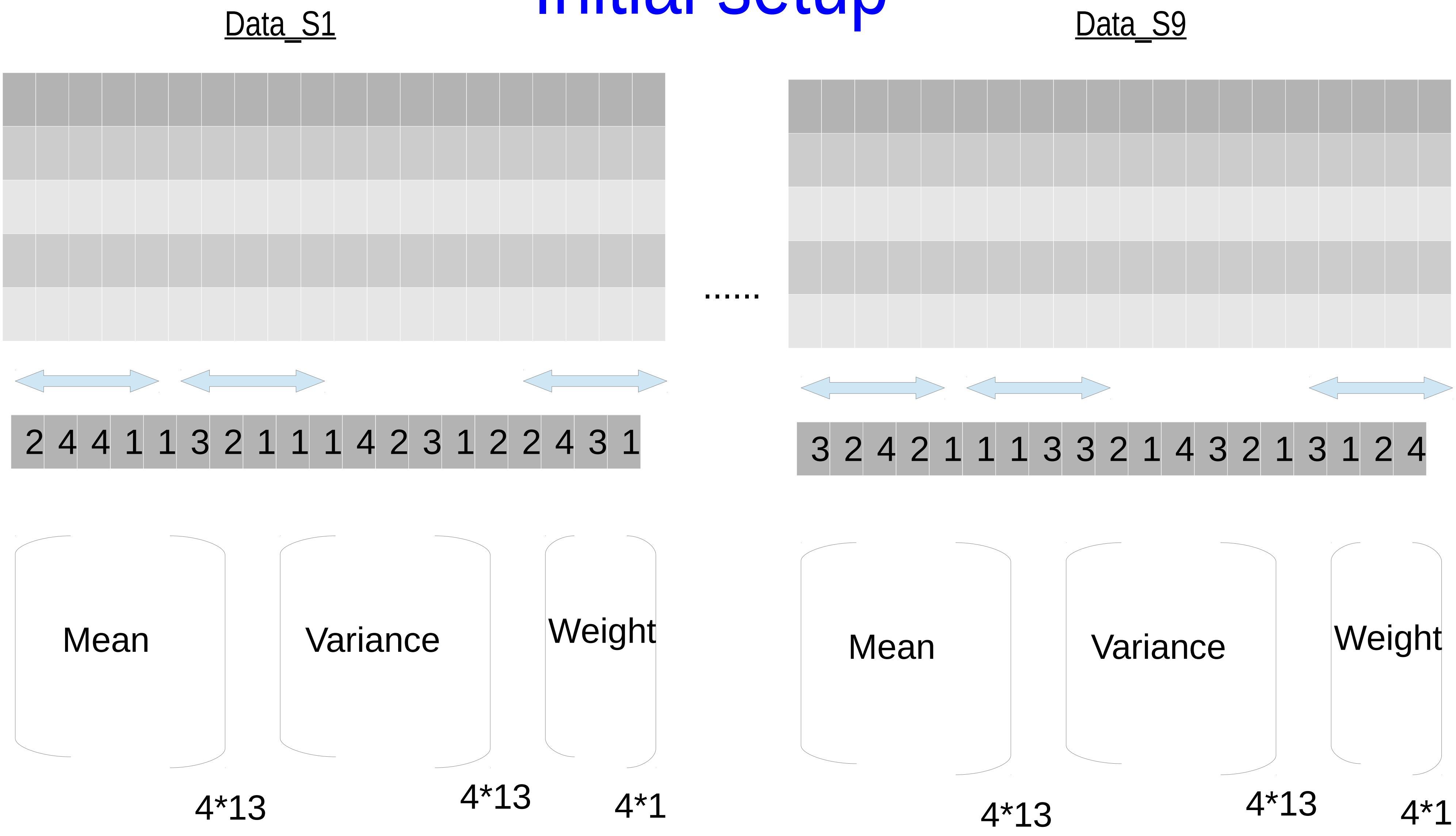


MFCC



Flat Start Initialization
Utterance--1
Utterance--2
Utterance--3
Utterance--4
Utterance--5
StateIndex
Frame Number
0
1
2
3
4
5
6
7
8
9
0
20
40
60
80
100
120
140
160

Initial setup



Forward Algorithm

1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N.$$

2) Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \le t \le T-1$$

$$1 \le j \le N.$$

3) Termination:

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

Backward Probabilities

1) Initialization:

$$\beta_T(i) = 1, \qquad 1 \le i \le N.$$

2) Induction:

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

$$t = T-1, T-2, \cdots, 1, \; 1 \le i \le N.$$

Viterbi Algorithm

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N$$

$$\psi_1(i) = 0.$$

2) Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \qquad 2 \le t \le T$$

$$1 \le j \le N$$

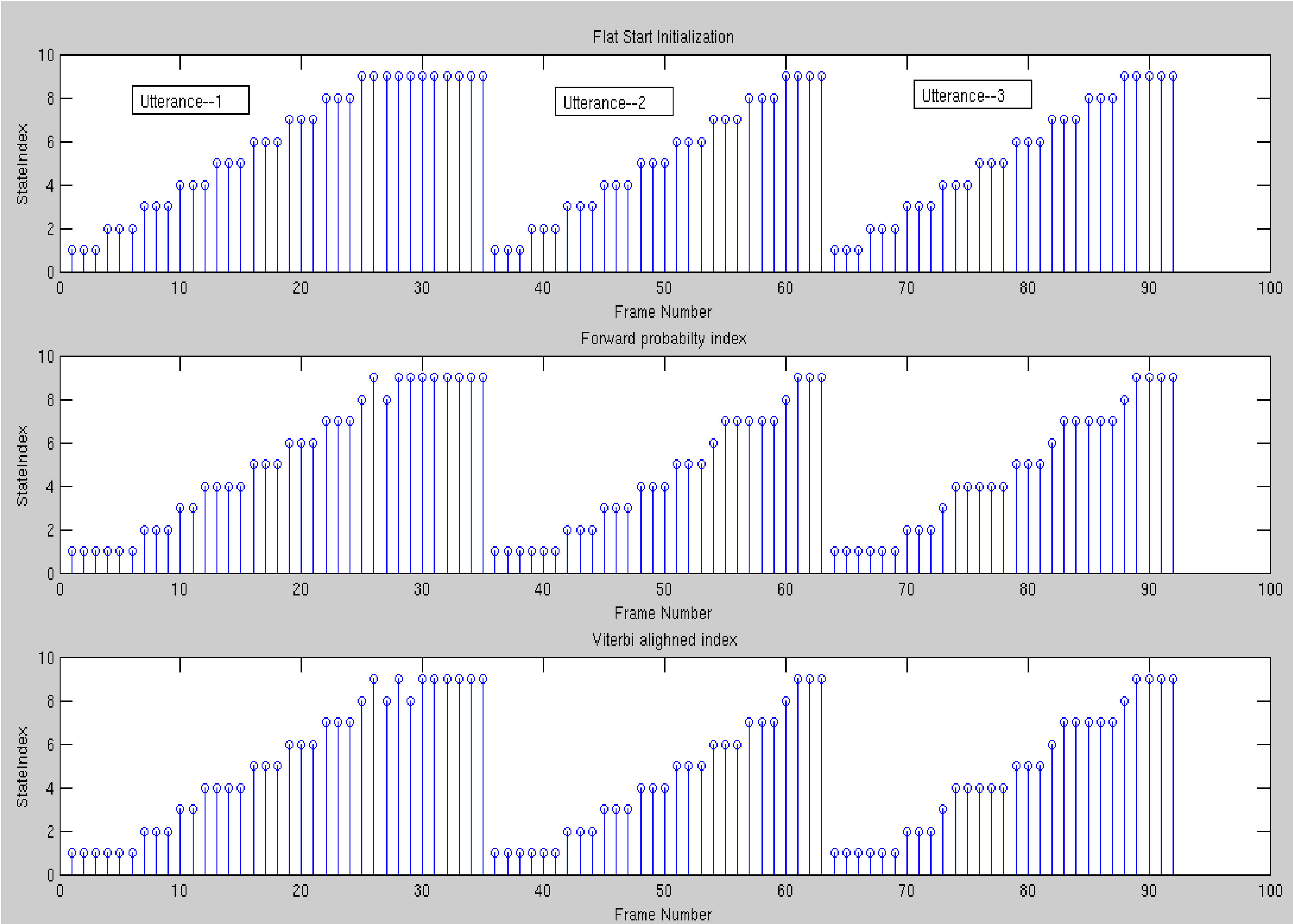
$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \qquad 2 \le t \le T$$

$$1 \le j \le N.$$

3) Termination:

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} [\delta_T(i)].$$



Flat Start Initialization
Utterance--1
Utterance--2
Utterance--3
StateIndex
Frame Number
Forward probabilty index
StateIndex
Frame Number
Viterbi alighned index
StateIndex
Frame Number

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

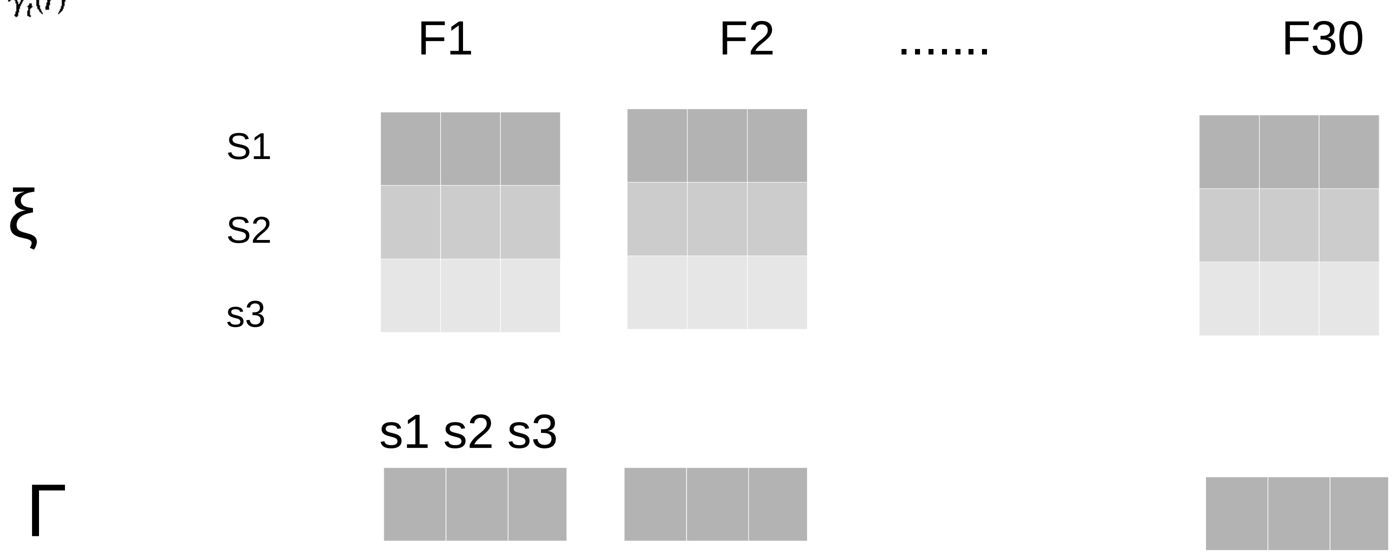
$$\xi_t(i, j) = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

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

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

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

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



A(S1,:) = zeeta(S1,:)./ [S1 S1 S1]

Editor - HMM_initial.m

Variable Editor - A_mat_updated

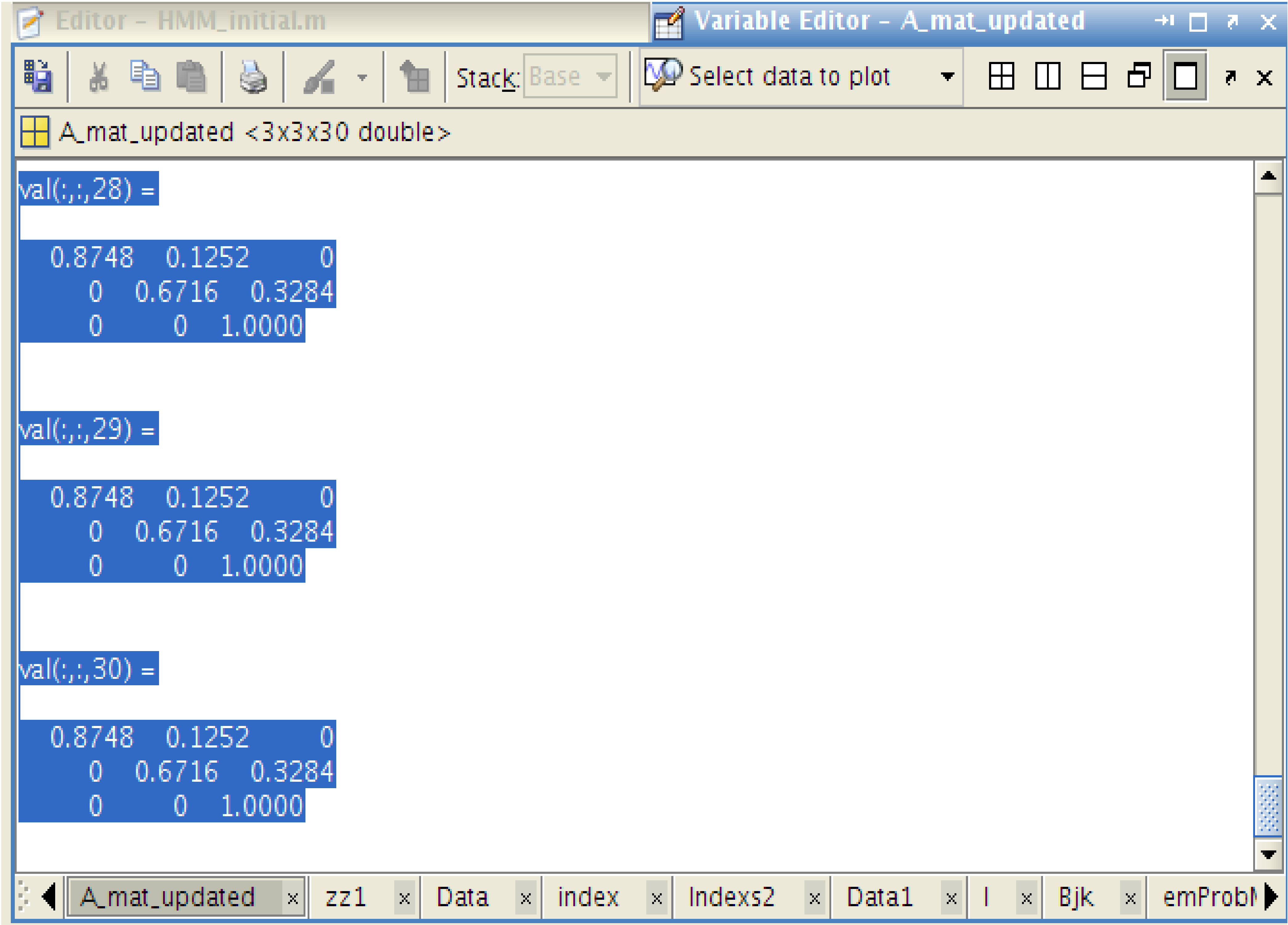
Stack: Base

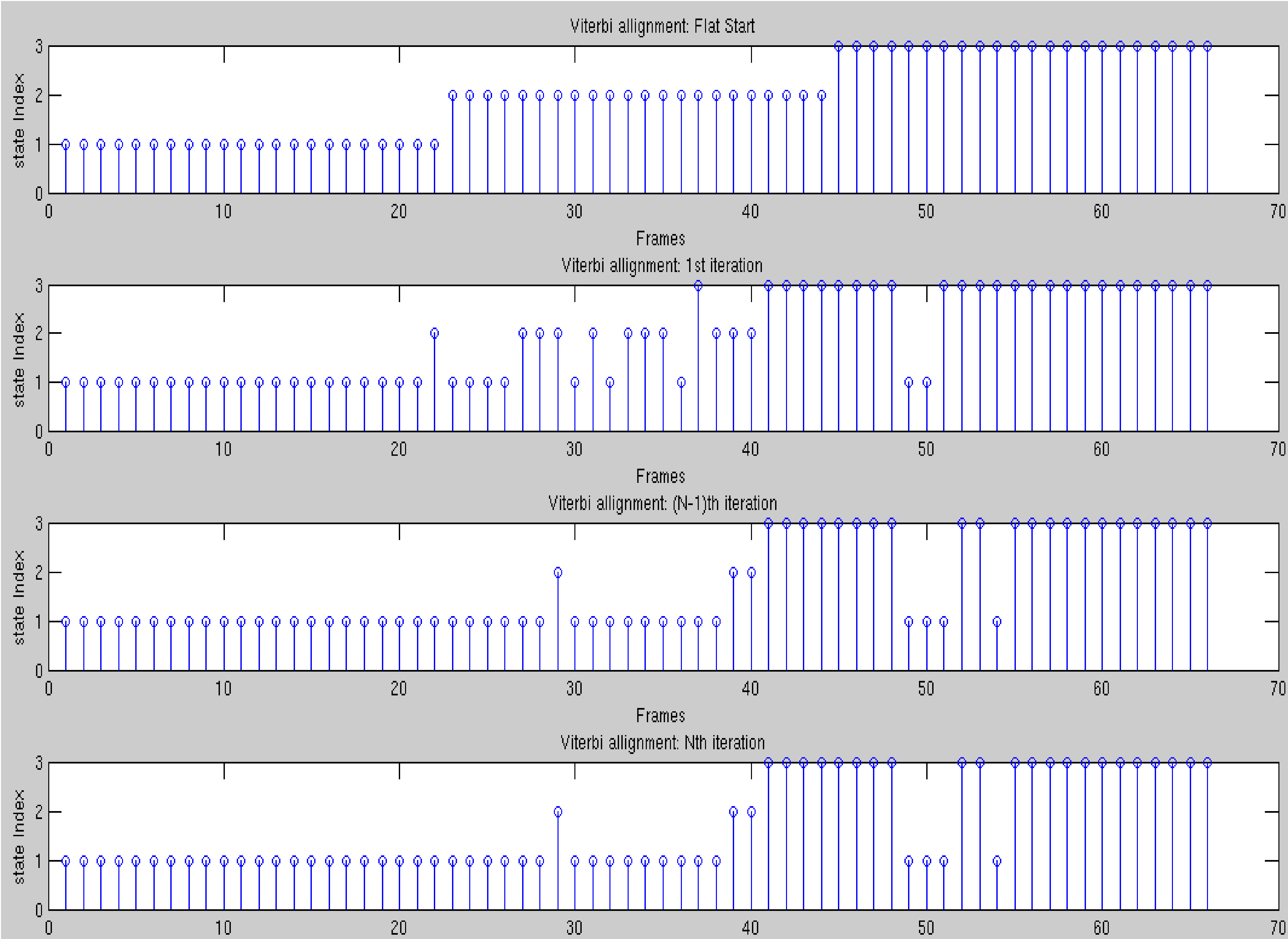
Select data to plot

A_mat_updated <3x3x30 double>

```
val(:,:,1) =

    0.8142    0.1858         0
         0    0.6706    0.3294
         0         0    1.0000


val(:,:,2) =

    0.8681    0.1319         0
         0    0.6716    0.3284
         0         0    1.0000


val(:,:,3) =

    0.8741    0.1259         0
         0    0.6716    0.3284
         0         0    1.0000
```

A1_mat_updated | A_mat_updated | zz1 | Data | index | Indexs2 | Data1 | I

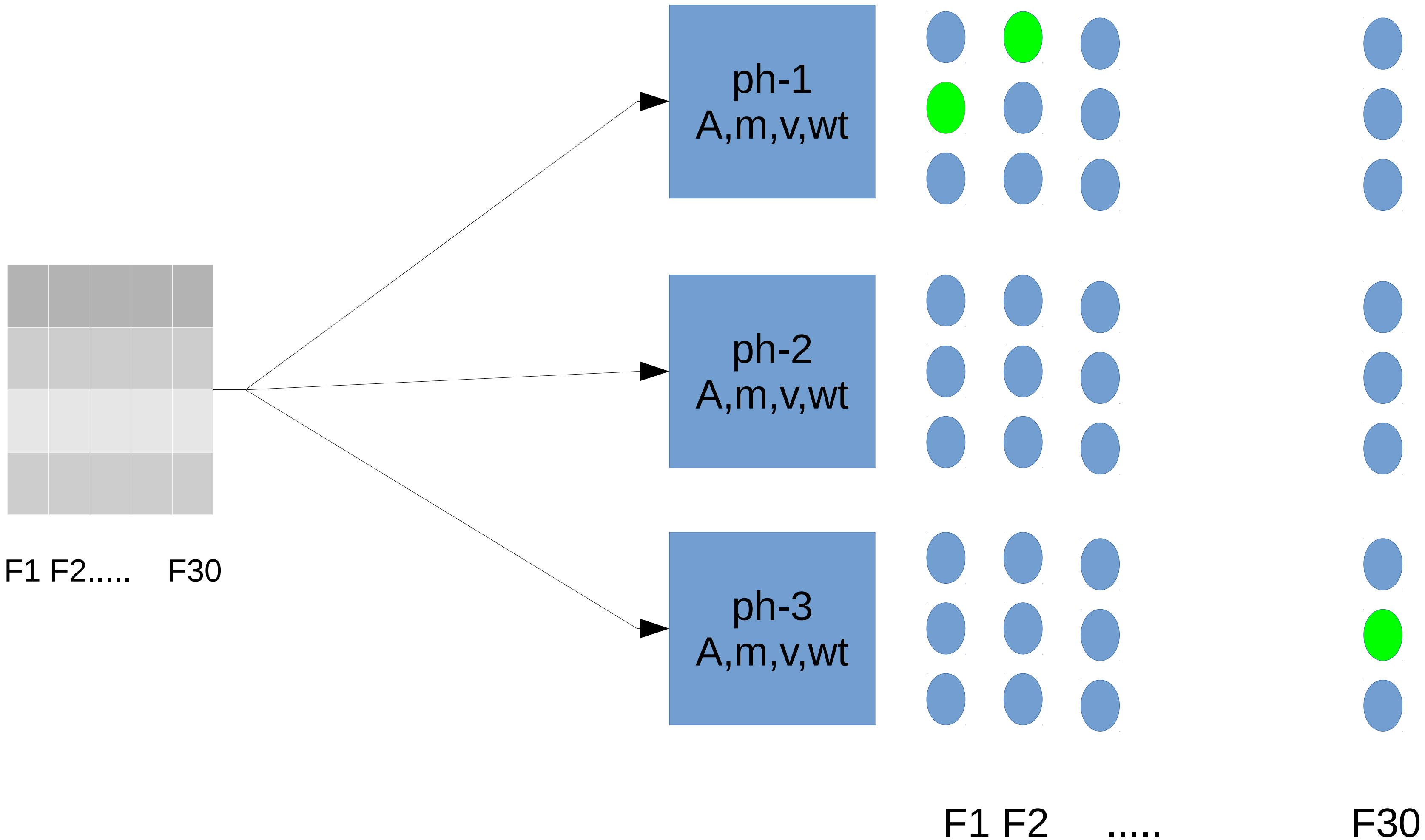

Editor - HMM_initial.m
Variable Editor - A_mat_updated
Stack: Base
Select data to plot
A_mat_updated <3x3x30 double>
val(:,:,28) =
0.8748 0.1252 0
0 0.6716 0.3284
0 0 1.0000
val(:,:,29) =
0.8748 0.1252 0
0 0.6716 0.3284
0 0 1.0000
val(:,:,30) =
0.8748 0.1252 0
0 0.6716 0.3284
0 0 1.0000
A_mat_updated
zz1
Data
index
Indexs2
Data1
I
Bjk
emProbM


Viterbi allignment: Flat Start
state Index
Frames
Viterbi allignment: 1st iteration
state Index
Frames
Viterbi allignment: (N-1)th iteration
state Index
Frames
Viterbi allignment: Nth iteration
state Index
0
1
2
3
0
10
20
30
40
50
60
70

Testing HMM

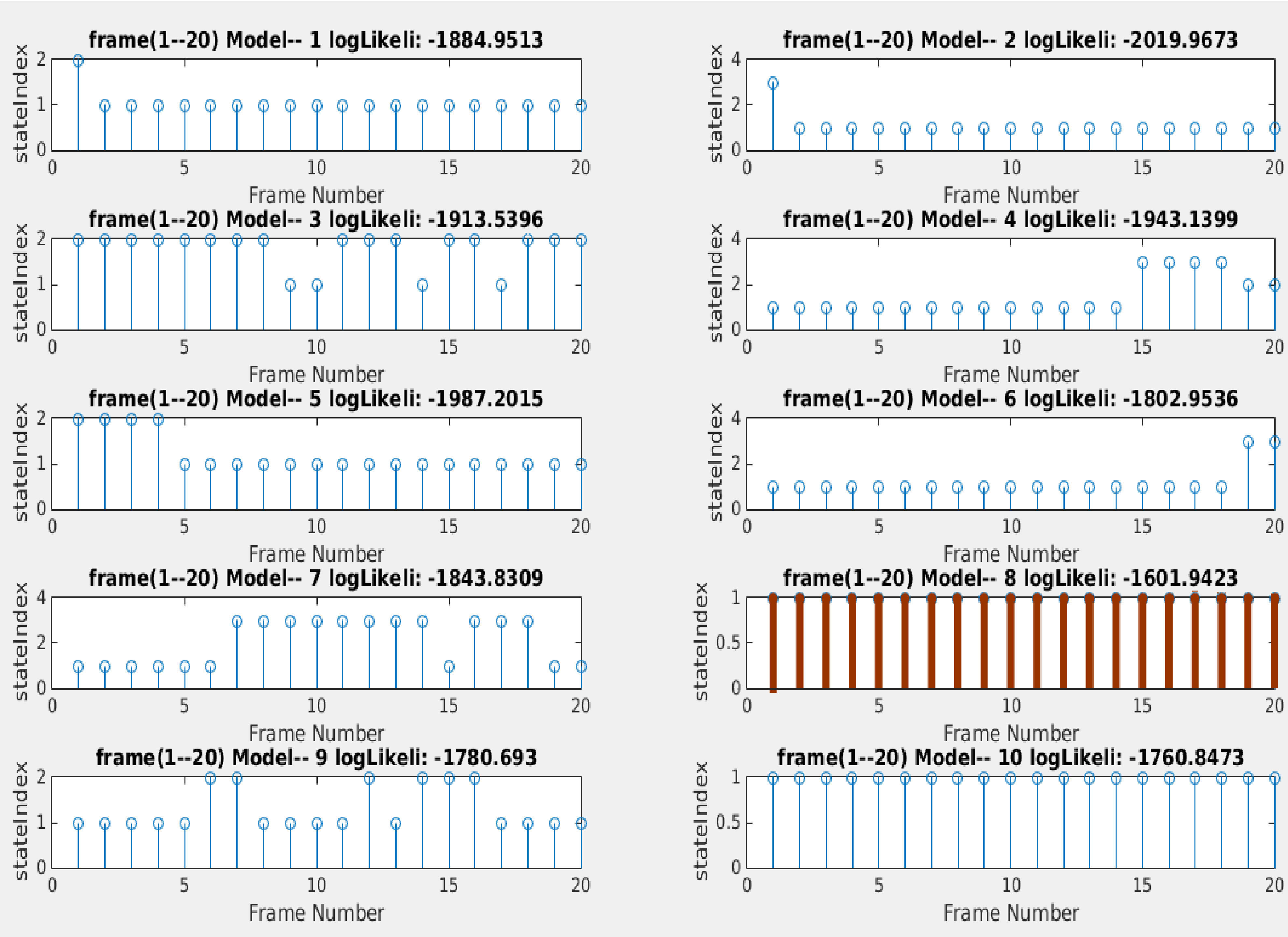
$$\left\{ \sum_{m=1}^{M_s} w_{ism} \mathcal{N}(\boldsymbol{o}_s; \boldsymbol{\mu}_{ism}, \Sigma_{ism}) \right\}$$

[A]*[b1 b2 b3]

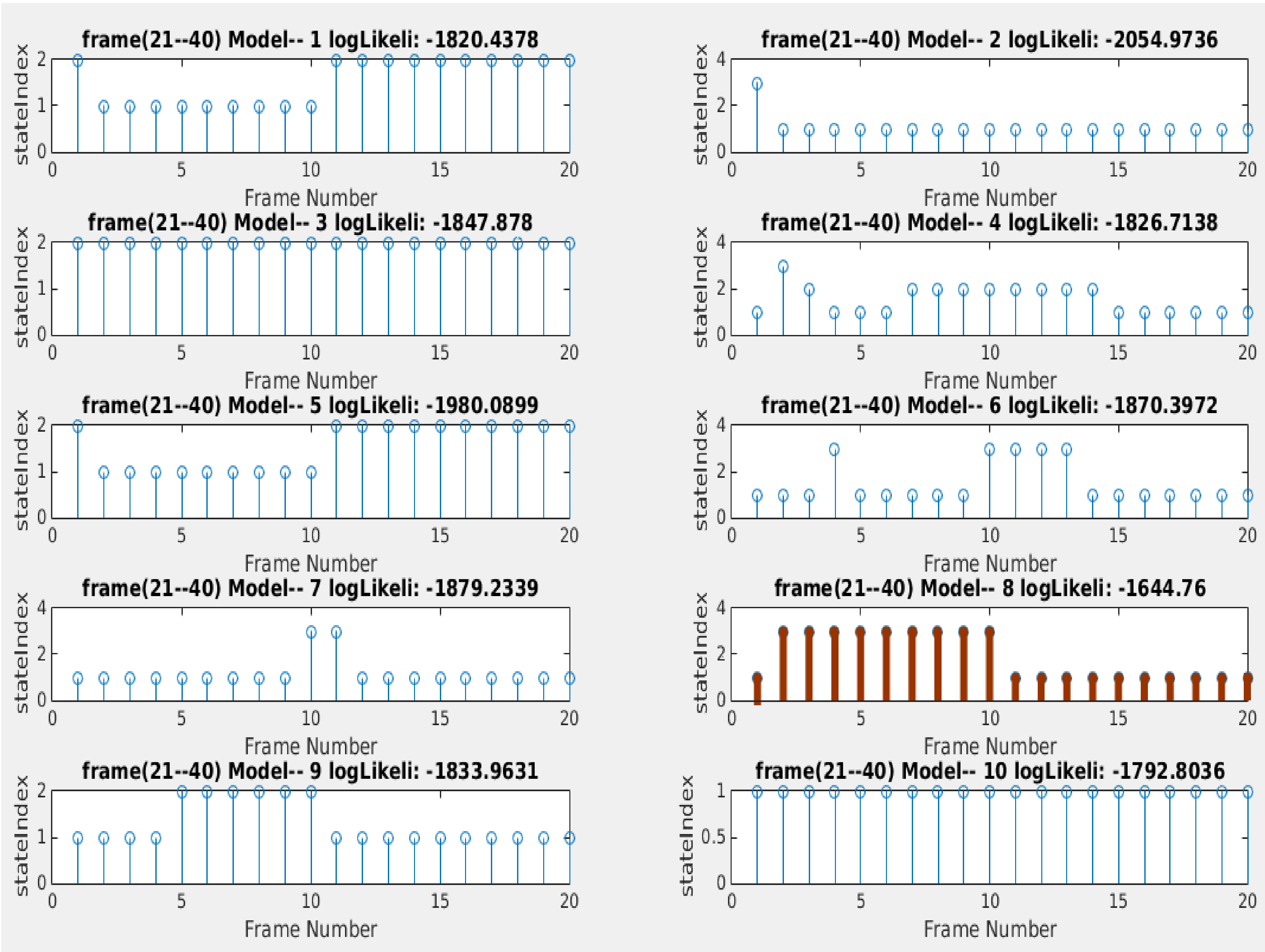


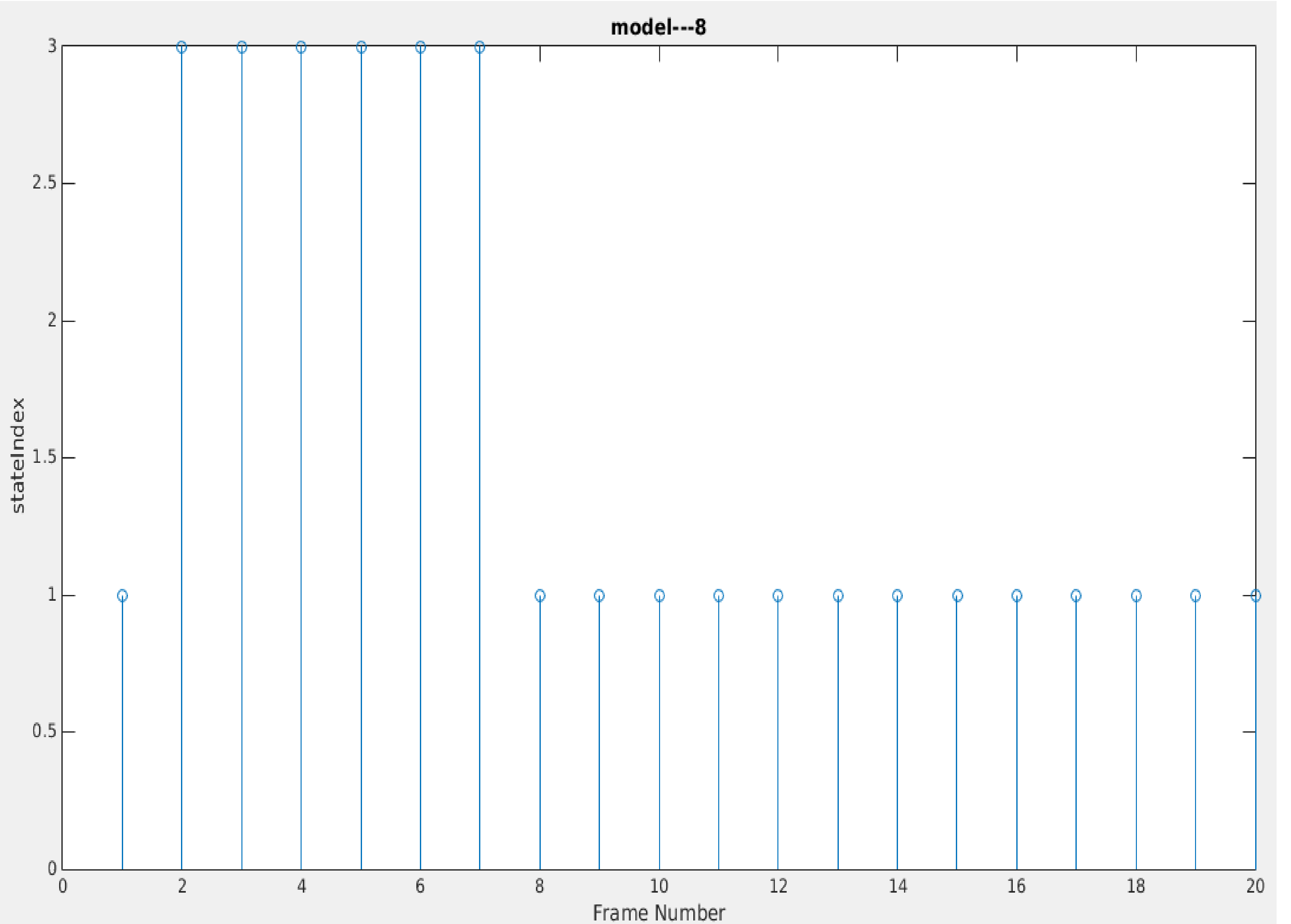
Calculating log-Likelihood from updated parameters

- Trained parameters for each state of a class:
  - Mean vector
  - Diagonal covariance matrix
  - Weighting factors for each Gaussian in a GMM
  - Updated Transition matrix (N*N)
  - priors
- Finding Likelihood:
  - $\alpha(:, t) = B(:, t) .* (A' * \alpha(:, t-1));$
  - $\text{logLikelihood} = \log(\text{sum}(\alpha(:, t)));$



frame(1--20) Model-- 1 logLikeli: -1884.9513
frame(1--20) Model-- 2 logLikeli: -2019.9673
frame(1--20) Model-- 3 logLikeli: -1913.5396
frame(1--20) Model-- 4 logLikeli: -1943.1399
frame(1--20) Model-- 5 logLikeli: -1987.2015
frame(1--20) Model-- 6 logLikeli: -1802.9536
frame(1--20) Model-- 7 logLikeli: -1843.8309
frame(1--20) Model-- 8 logLikeli: -1601.9423
frame(1--20) Model-- 9 logLikeli: -1780.693
frame(1--20) Model-- 10 logLikeli: -1760.8473
stateIndex
Frame Number


frame(21--40) Model-- 1 logLikeli: -1820.4378
frame(21--40) Model-- 2 logLikeli: -2054.9736
frame(21--40) Model-- 3 logLikeli: -1847.878
frame(21--40) Model-- 4 logLikeli: -1826.7138
frame(21--40) Model-- 5 logLikeli: -1980.0899
frame(21--40) Model-- 6 logLikeli: -1870.3972
frame(21--40) Model-- 7 logLikeli: -1879.2339
frame(21--40) Model-- 8 logLikeli: -1644.76
frame(21--40) Model-- 9 logLikeli: -1833.9631
frame(21--40) Model-- 10 logLikeli: -1792.8036
stateIndex
Frame Number


model---8
stateIndex
Frame Number
0
0.5
1
1.5
2
2.5
3
0
2
4
6
8
10
12
14
16
18
20

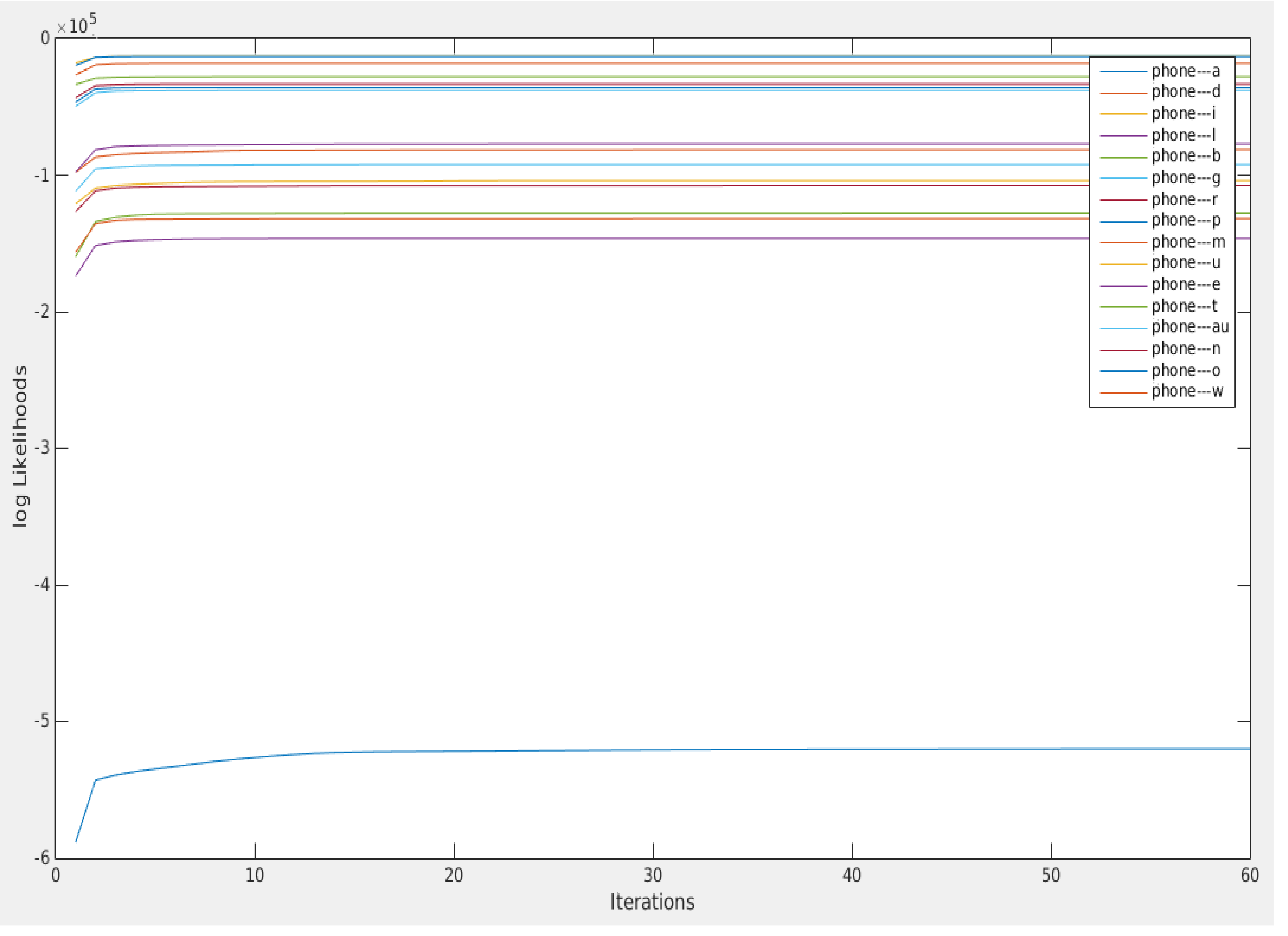
Building a 3-state Phone HMM

a d i l a b a d
a g r a
a l a p i
a m a l a p u r a m
a m b a l a
a m e t i
au r a n g a b a d
b a n g a l o r e
b e l g a u m
b i l w a r a

/a/	1
/d/	2
/i/	3
/l/	4
/b/	5
/g/	6
/r/	7
/p/	8
/m/	9
/u/	10
/e/	11
/t/	12
/au/	13
/n/	14
/o/	15
/w/	16

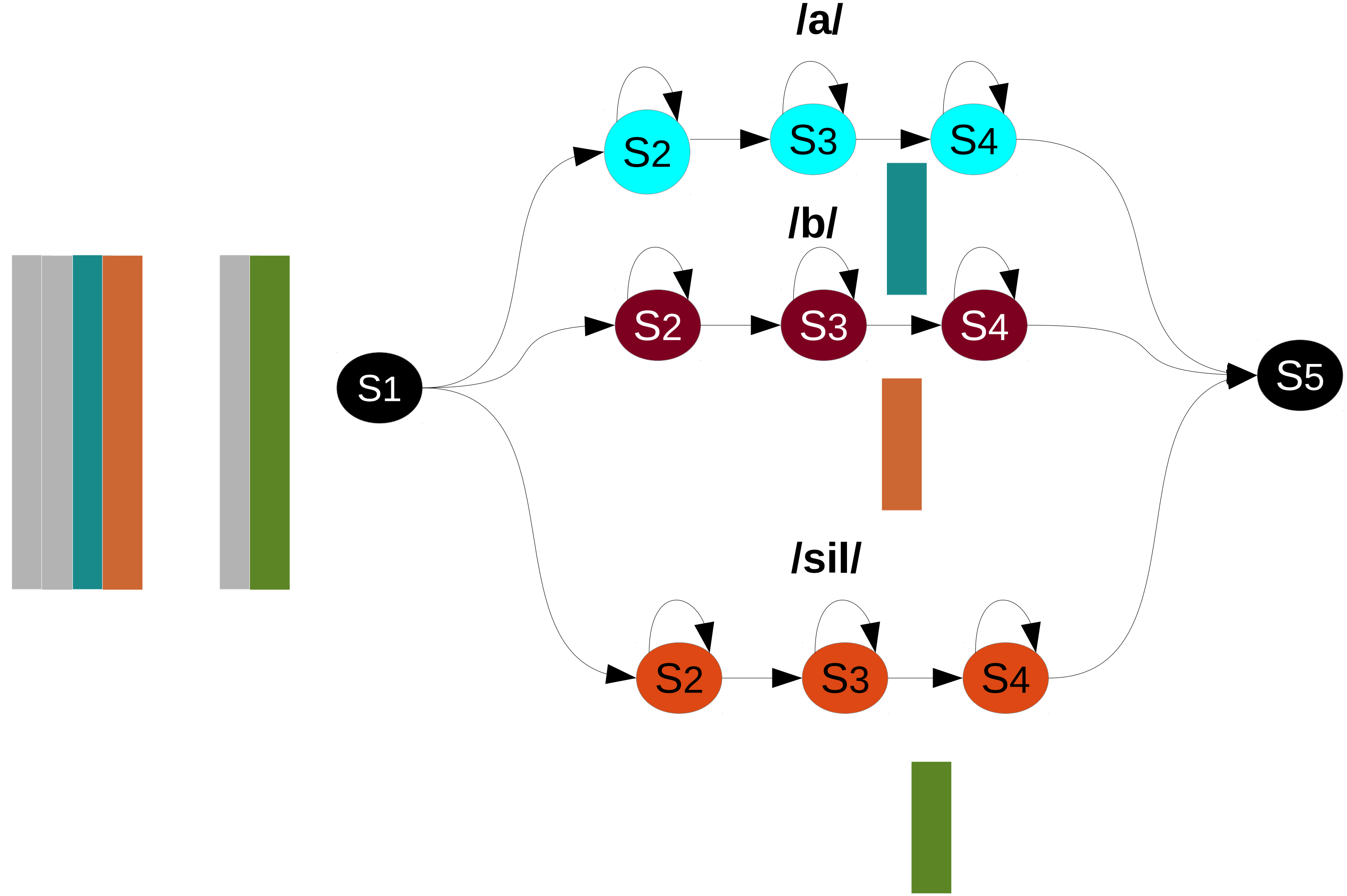
Steps:

- Flat start initialization----ex: 1 2 3 4 1 5 1 2
- Compute forward and backward probabilities
- Update the HMM parameters
- Stop training if the changes in HMM parameters is negligible

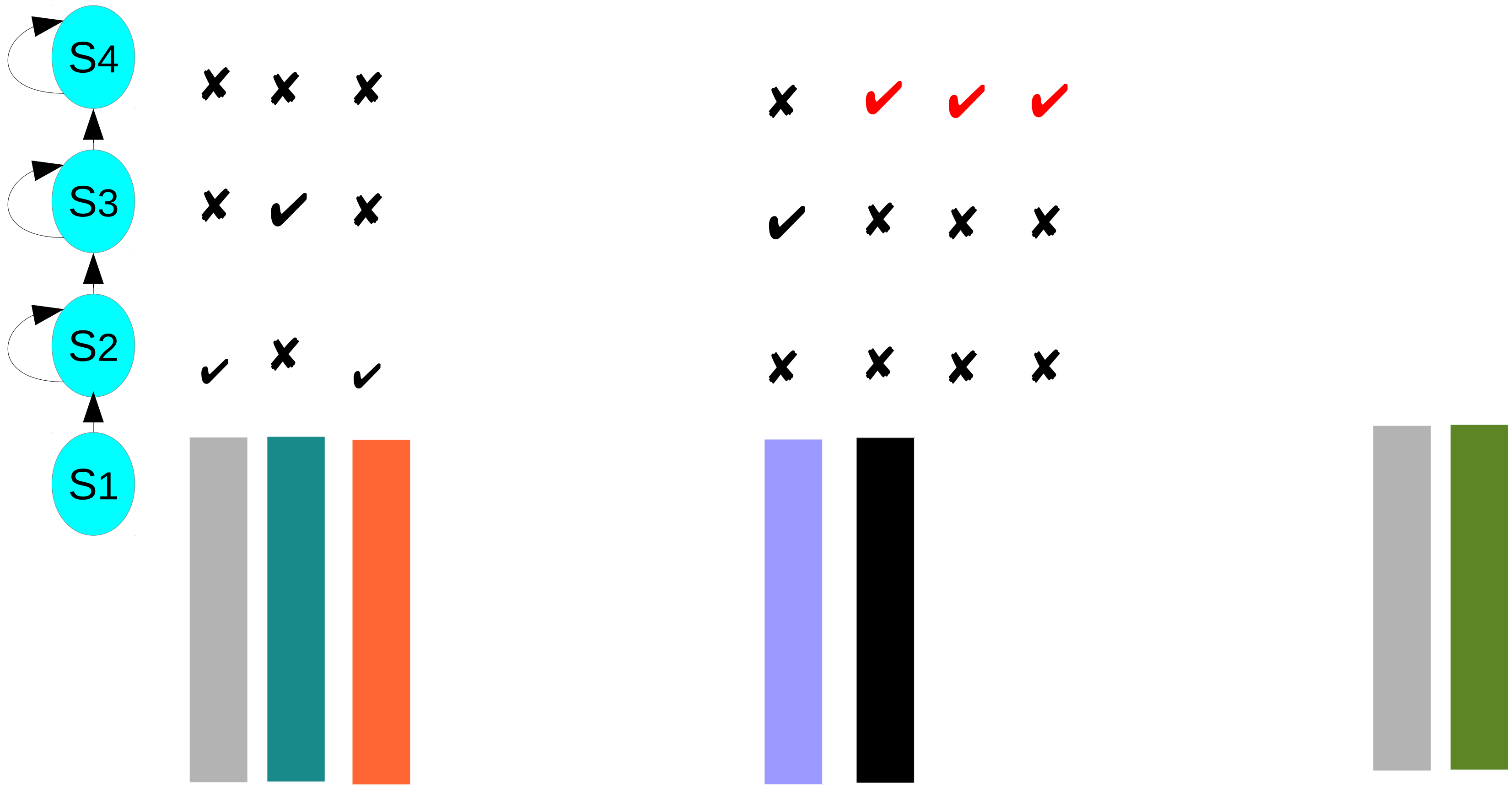


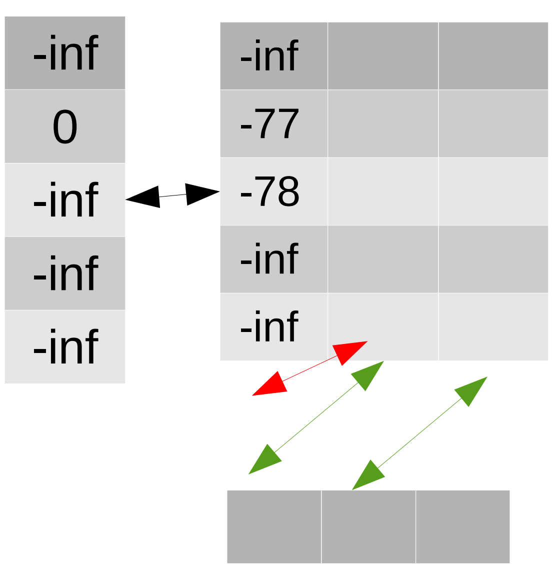
×10^5
0
-1
-2
-3
-4
-5
-6
log Likelihoods
0
10
20
30
40
50
60
Iterations
phone---a
phone---d
phone---i
phone---l
phone---b
phone---g
phone---r
phone---p
phone---m
phone---u
phone---e
phone---t
phone---au
phone---n
phone---o
phone---w

Analysing the break points



- Key idea:
  - Find the hidden state index where the final emitting state starts emitting the MFCC feature vectors
  - Analyze the corresponding viterbi path and its score w.r.t all trained models





- Identify the path with less significant Viterbi score
- Prune the Viterbi path by considering only the effect of maximum state sequence.

- t=2
-

$\delta_2(1)$
$\delta_2(2)$
$\delta_2(3)$
$\delta_2(4)$
$\delta_2(5)$

$\delta_2(j) = \max \begin{bmatrix} \delta_1(1).a_{1j} \\ \delta_1(2).a_{2j} \\ \delta_1(3).a_{3j} \\ \delta_1(4).a_{4j} \\ \delta_1(5).a_{5j} \end{bmatrix} \; b_3(F_j)$

At t=1, **p=2**

$$\begin{bmatrix} \delta_2(1) \\ \delta_2(2) \\ \delta_2(3) \\ \delta_2(4) \\ \delta_2(5) \end{bmatrix} = \begin{bmatrix} \delta_1(1).a_{p1} \\ \delta_1(2).a_{p2} \\ \delta_1(3).a_{p3} \\ \delta_1(4).a_{p4} \\ \delta_1(5).a_{p5} \end{bmatrix} \begin{bmatrix} b_3(F_p) \\ b_3(F_p) \\ b_3(F_p) \\ b_3(F_p) \\ b_3(F_p) \end{bmatrix}$$

<u>Token passing</u>

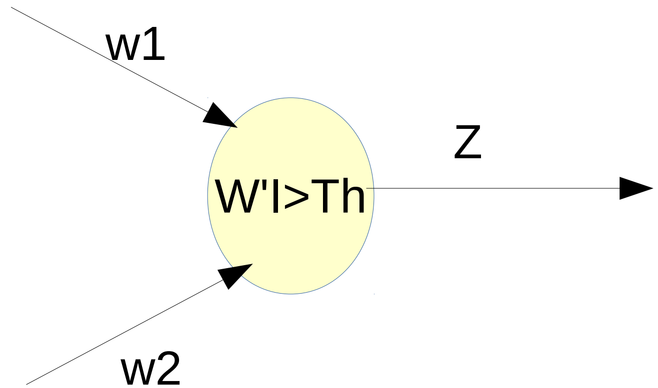
- Keep the best token arriving at each state
- Propagate these tokens to the next states
- Find the probabilty of the most likely state alignment

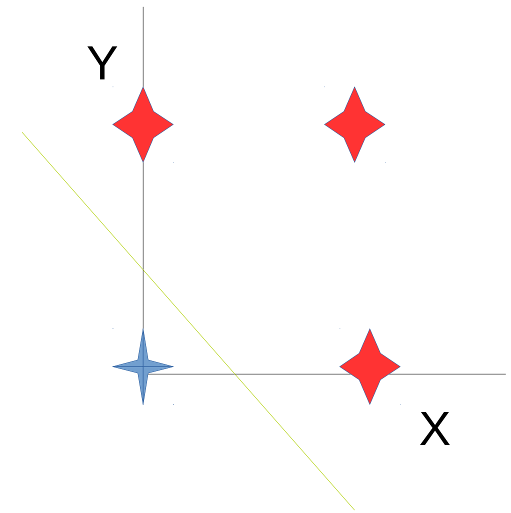
Practical Implementation of ANNs for Speech Applications

Introduction

OR-Gate

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1





0*w1+0*w2 < Th
0*w1+1*w2 >= Th
1*w1+0*w2 >=Th
1*w1+1*w2 >=Th

Th > 0
W2 >=Th
W1 >=Th
w1+w2 >=Th

y=mx+c
//m=w2/w1;
//c=Th;

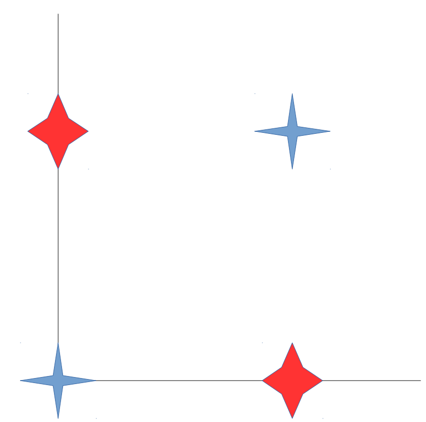
Soln : Th = 0.1;W = [0.15 0.2]

XOR-Gate

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

0*w1+0*w2 < Th
0*w1+1*w2 >= Th
1*w1+0*w2 >=Th
1*w1+1*w2 >=Th

Th > 0
W2 >=Th
W1 >=Th
w1+w2 < Th



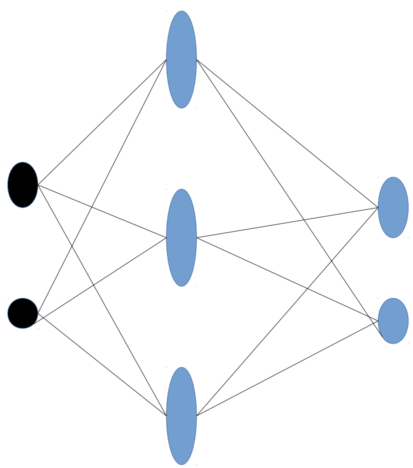
Need of Hidden Layer

- Project the lower dimensional data into higher one if they are not separable.
- These projections are via weight vectors.

Input

0	1	0	1
0	1	1	0

Class-1 Class-2



Desired Output

1	1	0	0
0	0	1	1

Class-1 Class-2

$W = [\ w_{11}, w_{12};$
$w_{21}, w_{23};$
$w_{31}, w_{32}]$

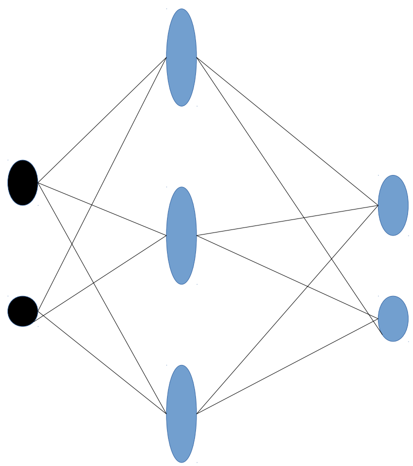
$W[3*2]*I[2*4] = I_h\ [3*4]$

Trained weights

Input

0	1	0	1
0	1	1	0

Class-1 Class-2



Desired Output

1	1	0	0
0	0	1	1

Class-1 Class-2

$W = [\quad w_{11}, w_{12};\ w_{21}, w_{23};\ w_{31}, w_{32}]$

$W[3*2]*I[2*4] = I_h\ [3*4]$

$\mathbf{W}_{ih}$ = [-6.793,12.214;
8.051,8.053;
12.208,-6.791];

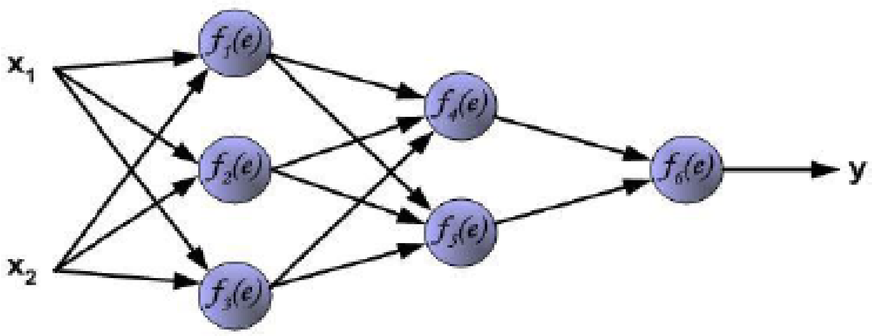
$\mathbf{W}_{ho}$ = [-11.58, 8.4, 8.4;
11.57, -8.39, -8.39];

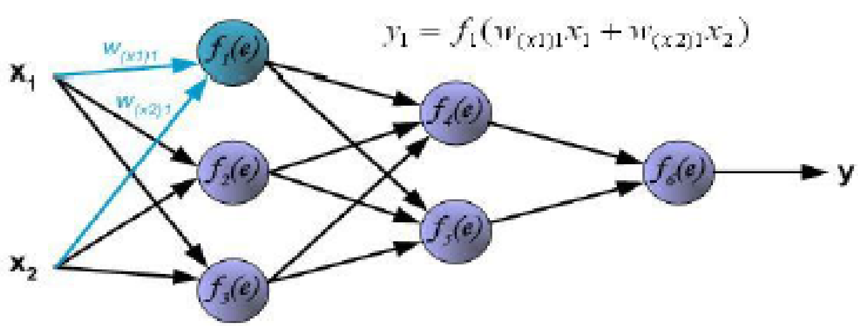
Actual output
(after training)

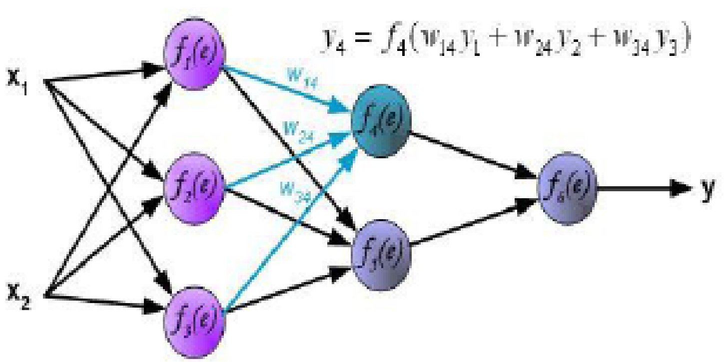
0.93	0.99	0.04	0.04
0.06	0.01	0.96	0.96

MSE = sum(Desired-Actual).^2
=9.9*10e-4

Illustration of Back-Propagation learning algorithm

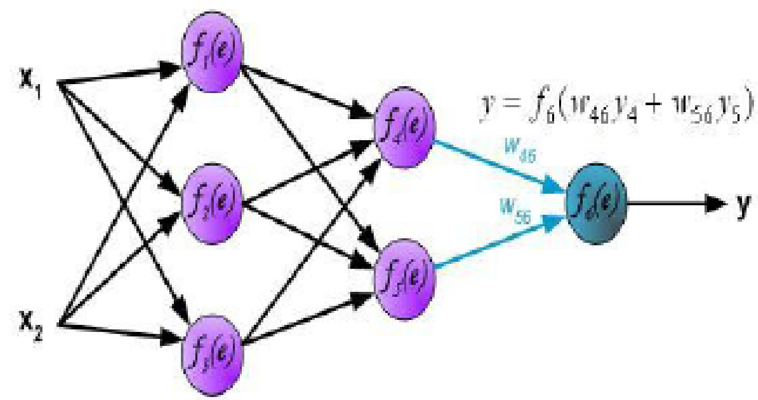


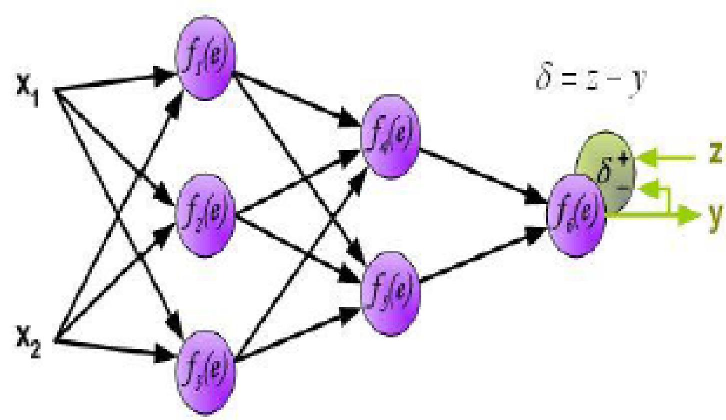




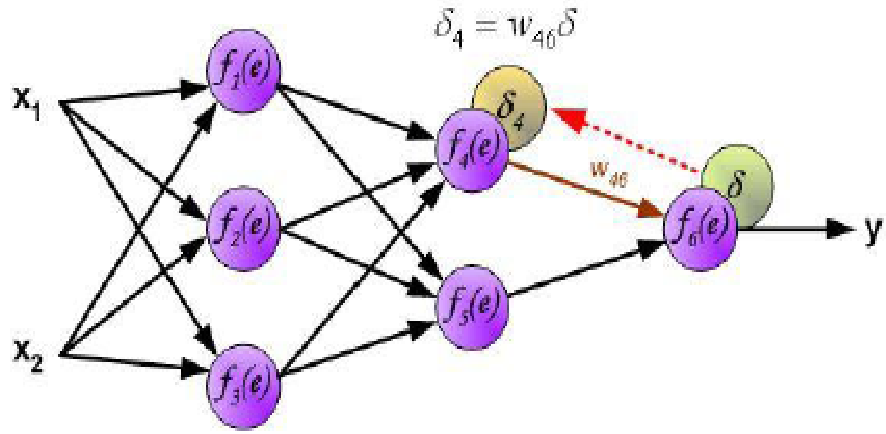
Sigmoid activation fn.

$f(x)=1/(1+exp(-x))$

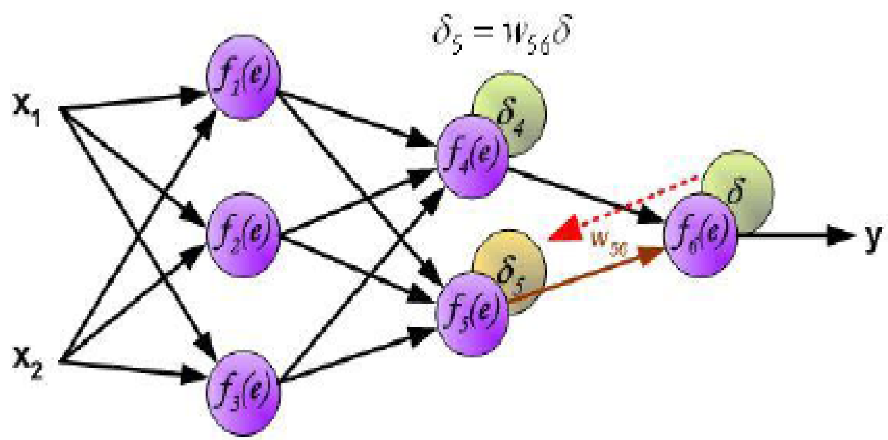




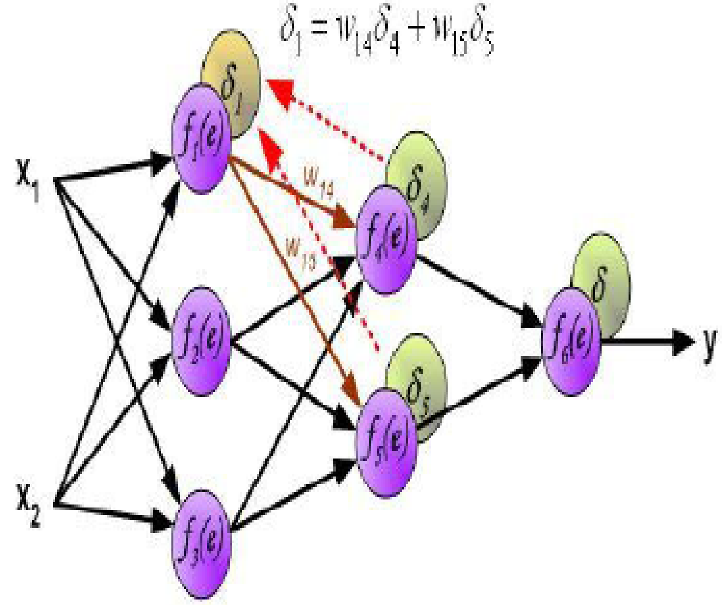
http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html



$\delta_4 = w_{46}\delta$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_4
δ
w_{46}
y



$\delta_5 = w_{56}\delta$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_4
δ_5
δ
w_{56}
y

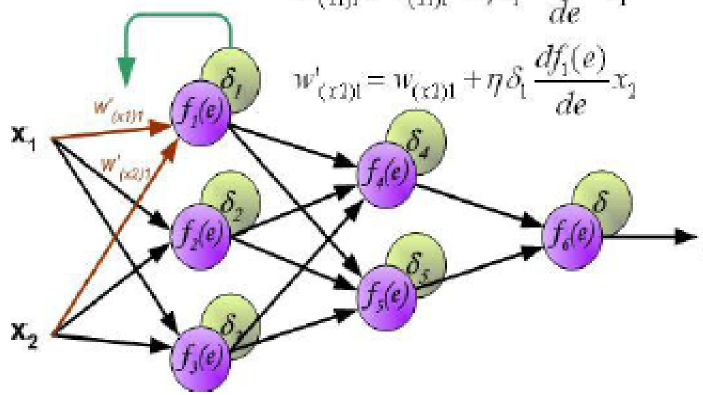


$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_1
δ_4
δ_5
δ
w_{14}
w_{15}
y

- Weight Update: Delta learning rule

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

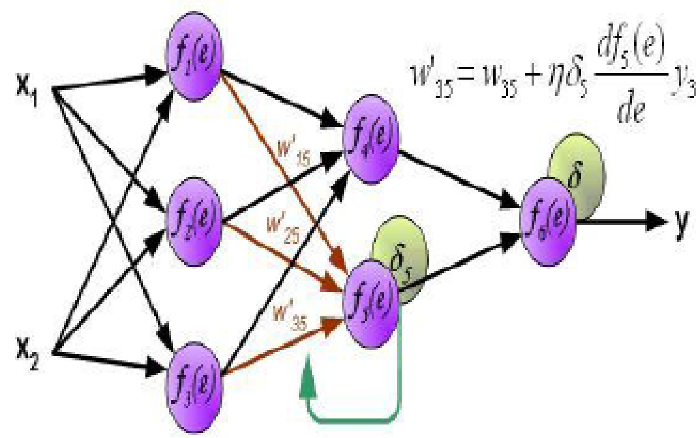
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

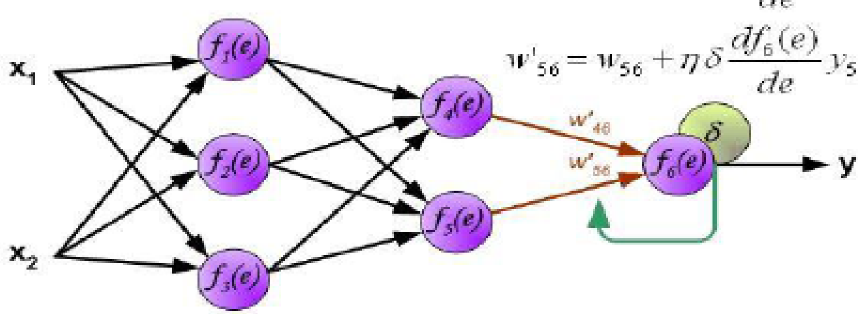
$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$

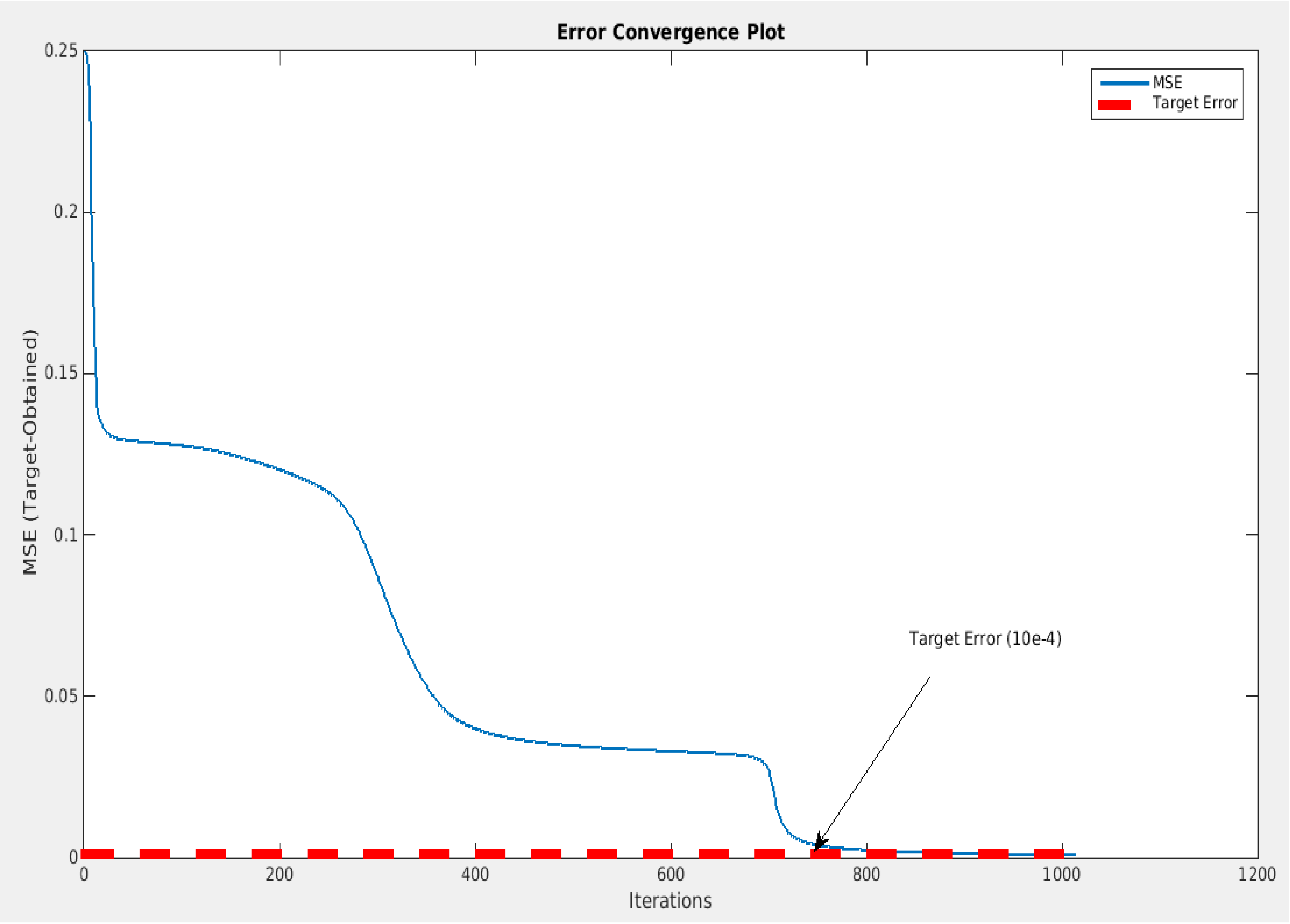


$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$

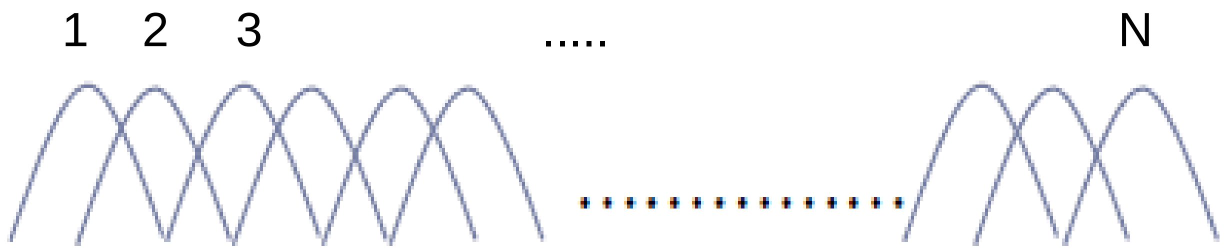


- Stopping Criteria: Mean square error <= Goal error

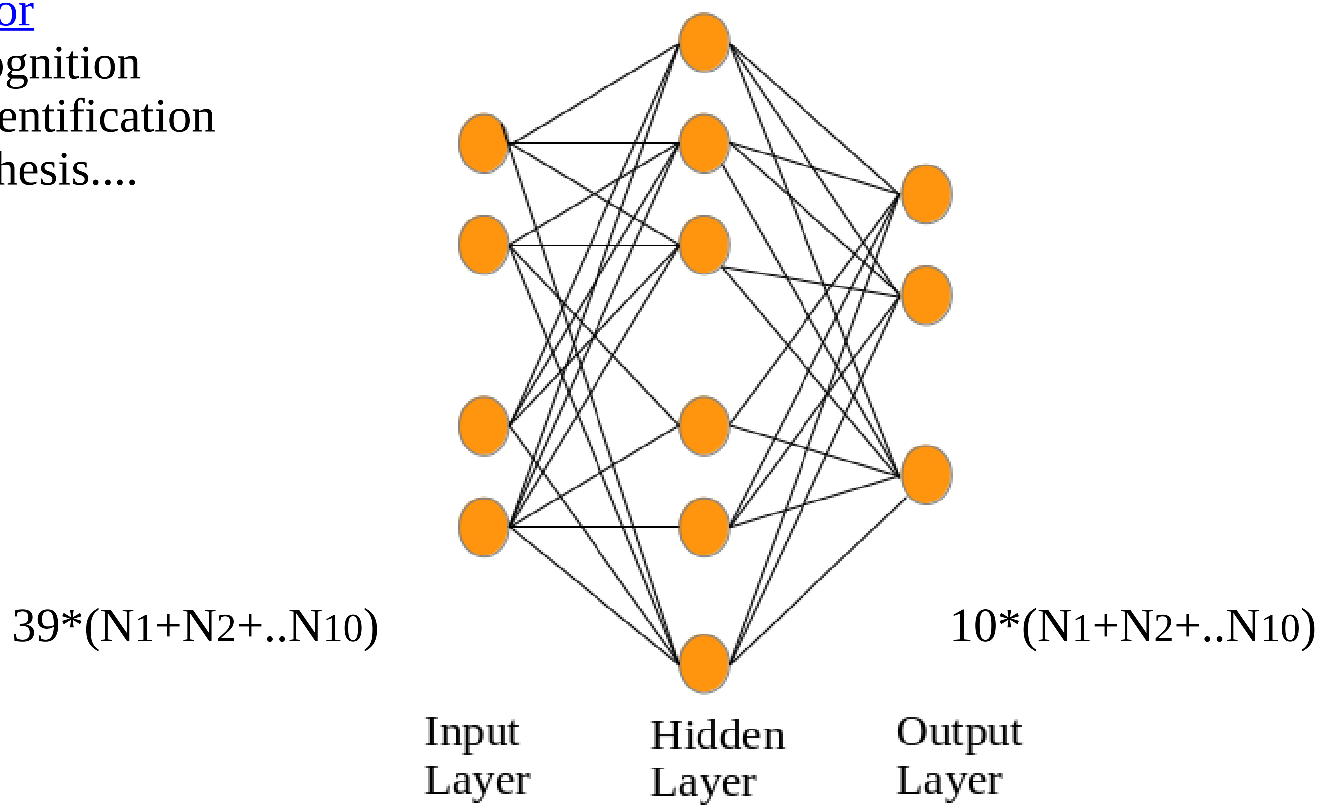


Error Convergence Plot
MSE
Target Error
Target Error (10e-4)
MSE (Target-Obtained)
Iterations
0
0.05
0.1
0.15
0.2
0.25
0
200
400
600
800
1000
1200

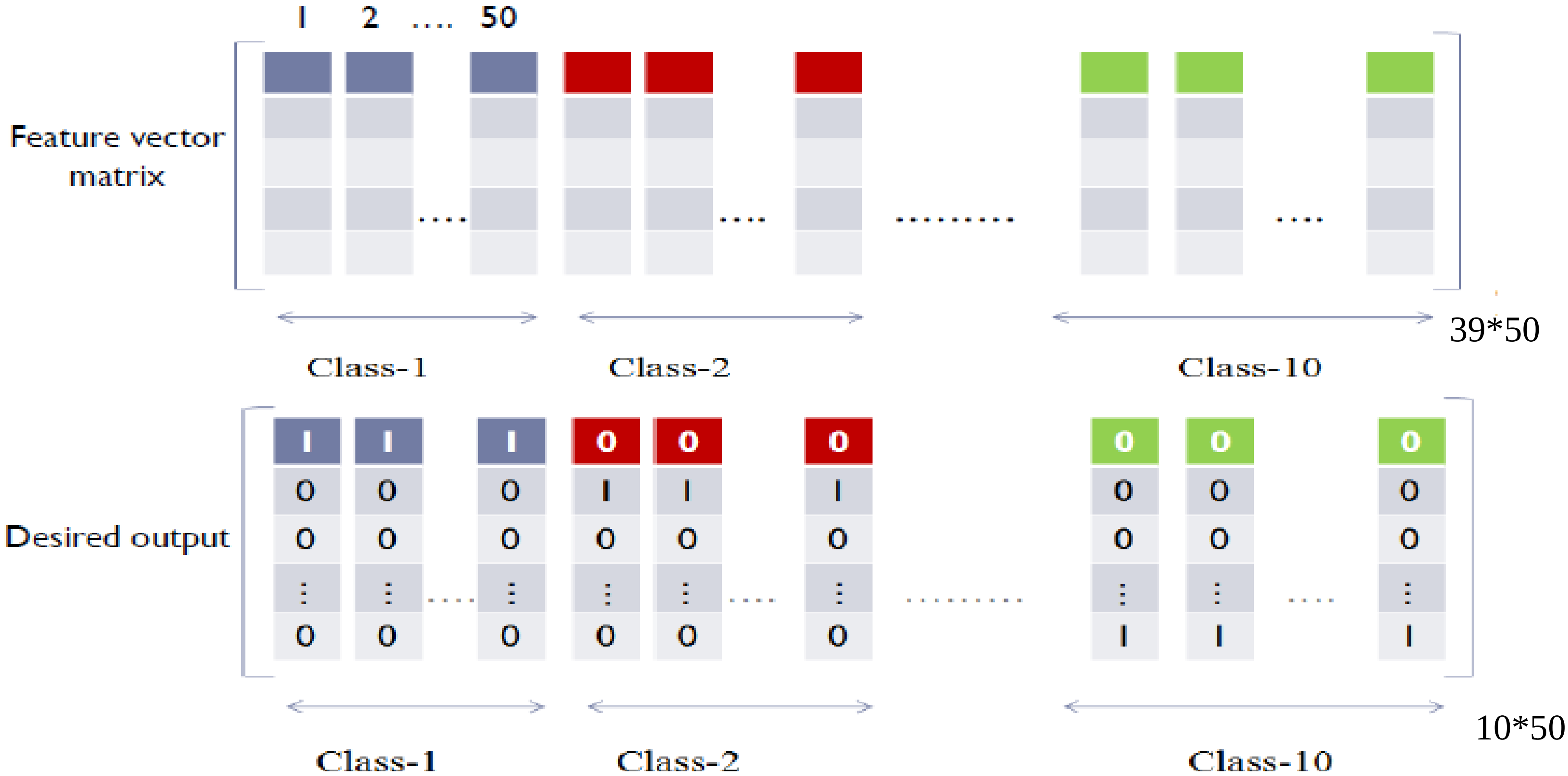
ANN for Speech Applications



<u>Applicable for</u>
Speech Recognition
Language Identification
Speech Synthesis....



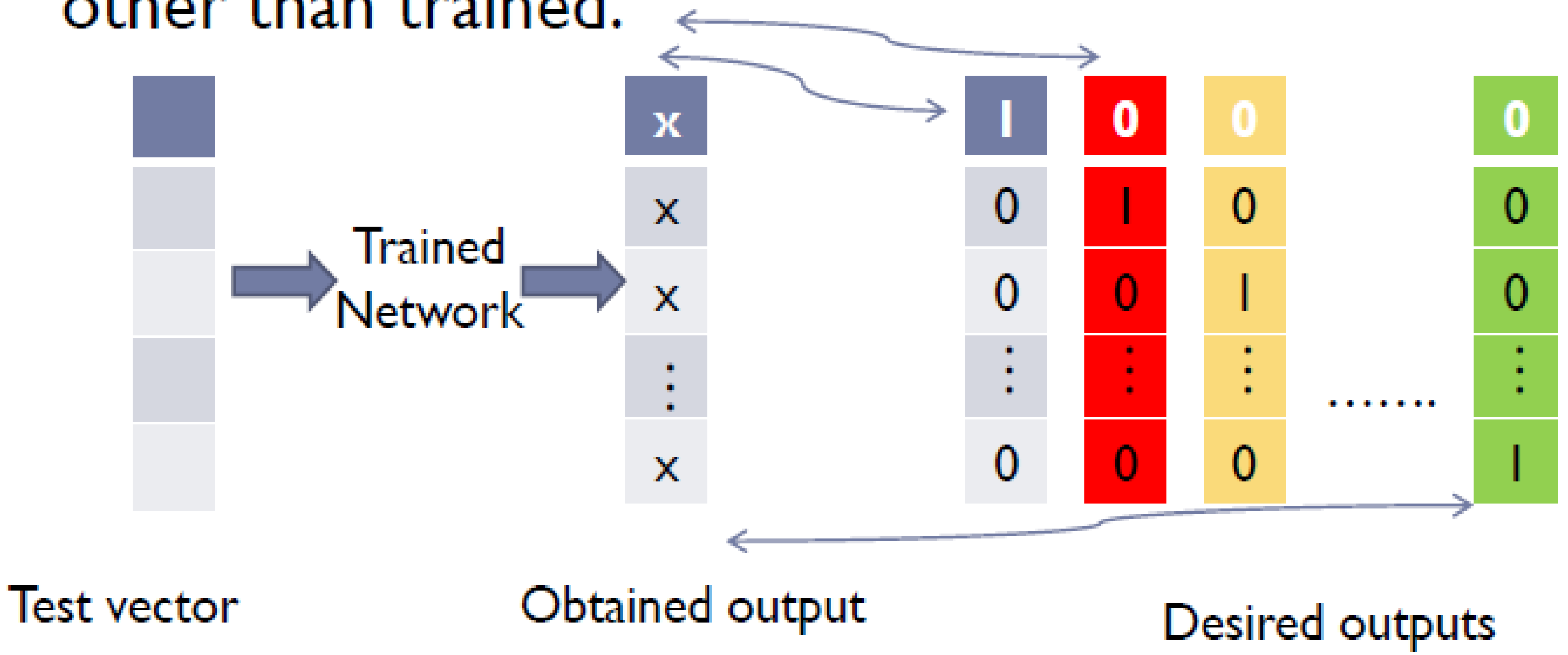
Training Feed forward Neural networks



- Learning rate=0.001
- Min. MSE=0.0001

Testing FFNN

- Weights are updated implying that the network has learnt.
- Check the algorithm efficiency by feeding a new input other than trained.



Comparing GMMs and ANNs

GMMs	ANNs
$\hat{w} = \underset{w}{\operatorname{argmax}}\, p(w \mid \mathbf{x}) = \underset{w}{\operatorname{argmax}}\, p(\mathbf{x} \mid w)p(w)/p(\mathbf{x})$	$y_j = logistic(x_j) = \frac{1}{1+e^{-x_j}}, \qquad x_j = b_j + \sum_i y_i w_{ij},$
▪Maximize the Likelihood score	▪Maximize the posterior probabilities
▪Needs appropriate choice of Mixture components	▪Choice of nodes and layers plays a vital role

Future Work

- To create HMM-ANN baseline
- HMM State Transitions

ANN posteriors

Viterbi Aligned Result

HMM-ANN Baseline

Deep Neural Networks(DNNs)

[Hinton et. al, *Neural Computation*, 2006]

- DNNs or Deep Belief Networks (DBNs) are Multi layer perceptrons with many hidden layers but the difference exist in the training phase.

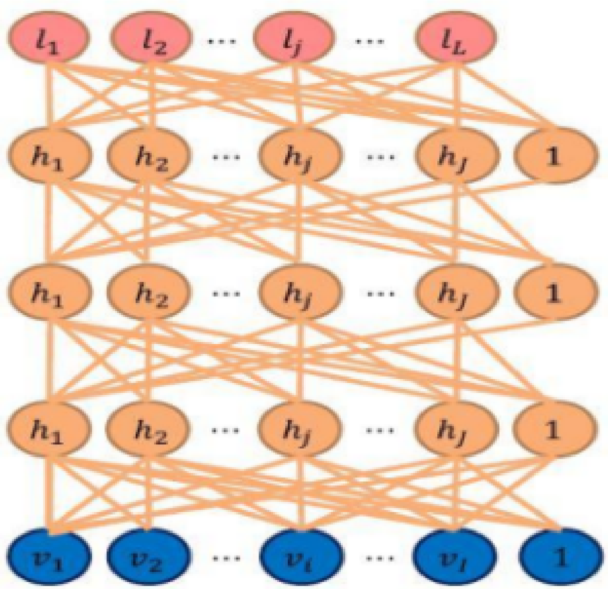


Fig. DBN/DNN Architecture

RBM-4

RBM-3 Restricted Boltzmann Machines

RBM-2

RBM-1

- G.Hinton discovered learning strategy for Multilayer Perceptrons

a) Pre-train each layer from bottom to top
b) Each pair of layers is an Restricted Boltzmann Machine(RBM)
c) Jointly fine-tune all layers using back-propagation

Restricted Boltzmann Machines

[Hinton et. al, *Neural Computation*, 2006]

- DNNs are formed by stacking several RBMs one upon another.

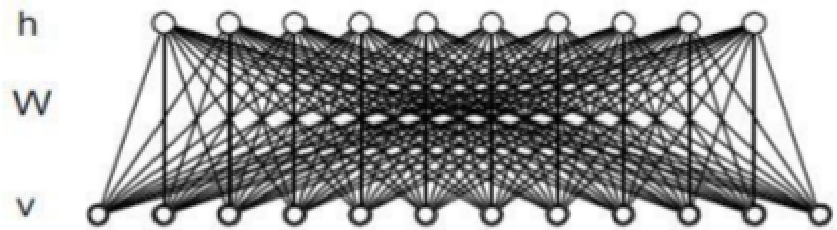


Fig. Restricted Boltzmann Machine (RBM)

Bernoulli-Bernoulli RBM

$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{I} w_{ij}\, v_i + a_j\right),$$

$$p(v_i = 1|\mathbf{h};\theta) = \sigma\left(\sum_{j=1}^{J} w_{ij}\, h_j + b_i\right),$$

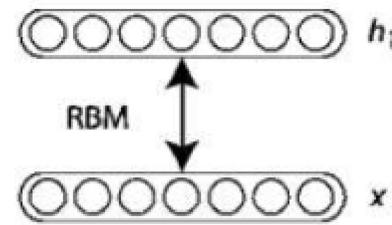
Gaussian-Bernoulli RBM

$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{I} w_{ij}\, v_i + a_j\right),$$

$$p(v_i|\mathbf{h};\theta) = \mathcal{N}\left(\sum_{j=1}^{J} w_{ij}\, h_j + b_i, 1\right),$$

$$\sigma(x)=1(1+exp(-x))$$

Unsupervised Pre-training



- $v=x_1$ is the input distribution.
- Find activations at the hidden layer

$$p(h_j = 1 \mid v;\theta) = \sigma(\sum_{i=1}^{V} w_{ij} v_i + a_j)$$

- Convert activation probabilities to binary representation, h2
- Compute the activations at the visible unit given the binary data at the hidden unit.

$$p(v_i = 1 \mid h;\theta) = N(\sum_{j=1}^{H} w_{ij} h_j + b_i)$$

- Again the activation probabilities at the hidden units given the new input distribution are found, $P(h_2=1|x_2)$.
- The weight update rule for 1^{st} iteration as described by the above procedure is given as

$$W=W+(h_1 x_1^T - P(h_2=1|x_2) x_2^T)$$

Contrastive Divergence

Stacking RBMs

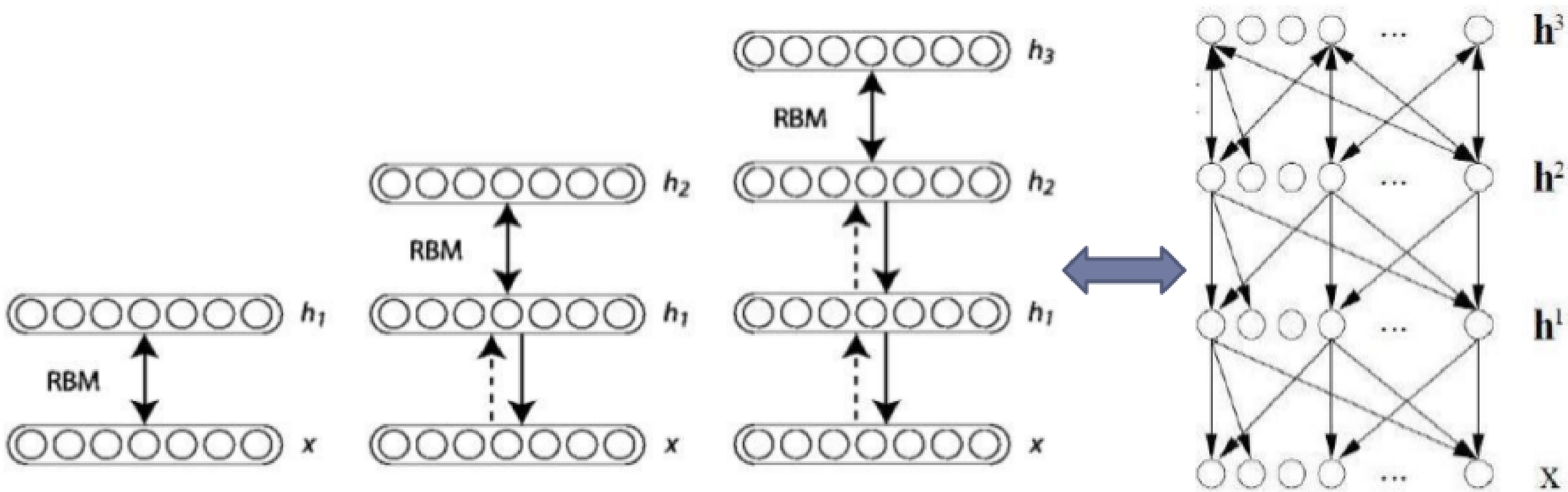
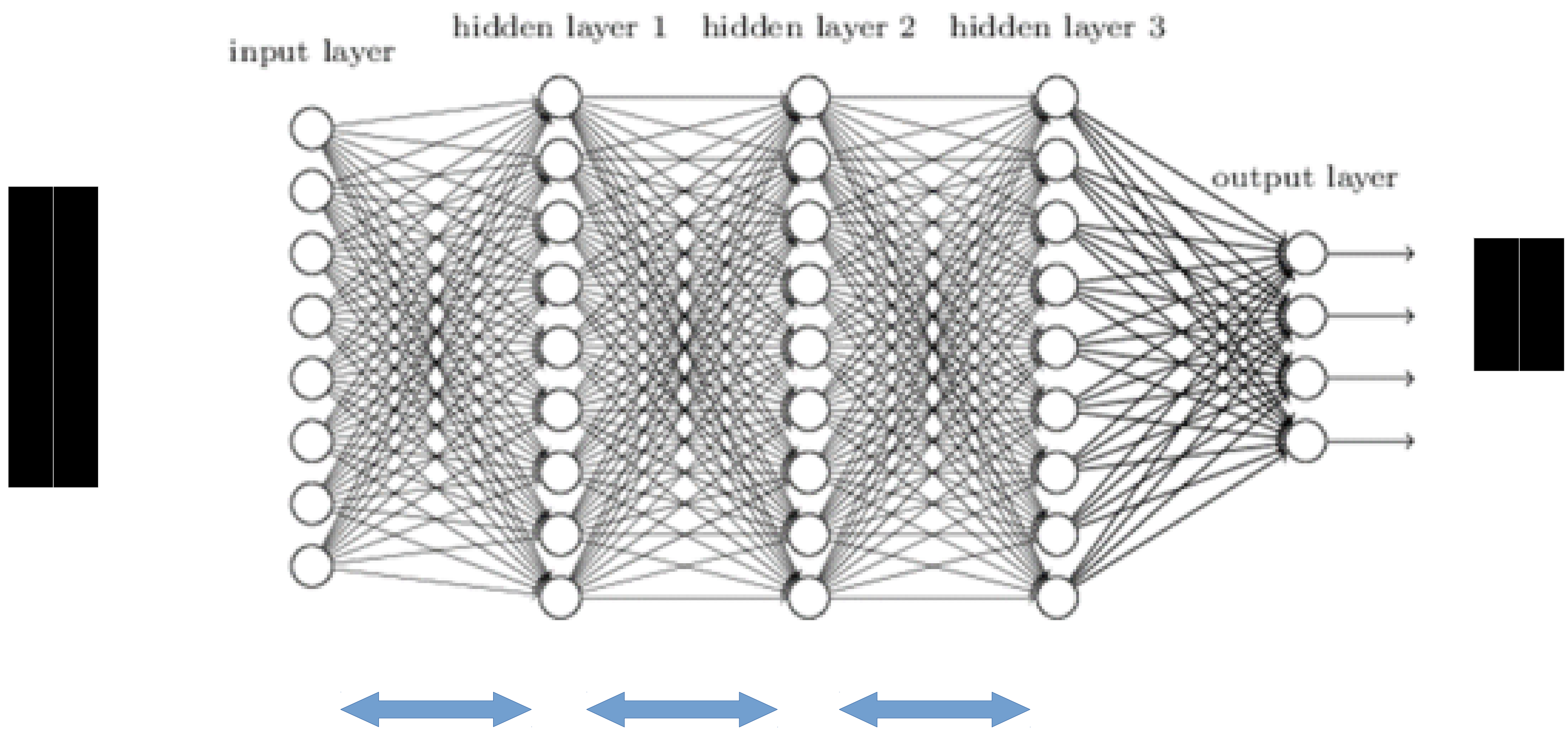


Fig. Stacking RBMs to form a DBN

- The activation probabilities of hidden units are used as the visible data for the next layer.
- This process is repeated to create other layers.
- Now Back-propagation algorithm is applied to adjust the DBN weights (fine-tuning phase)

DNN

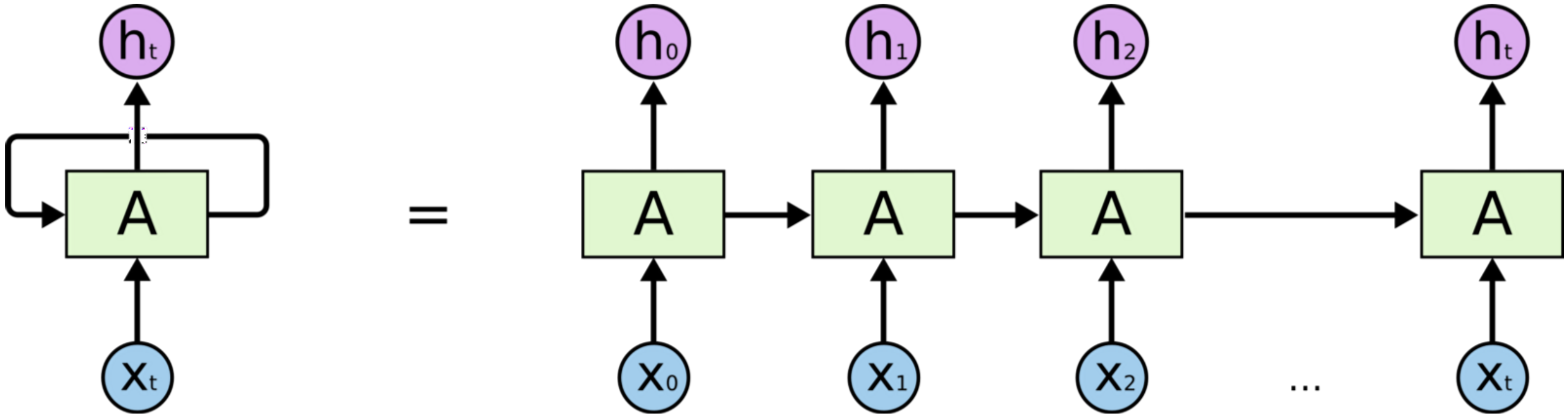


RBM

Training:

- Unsupervised Pre-training
- Supervised Fine tuning

RNN

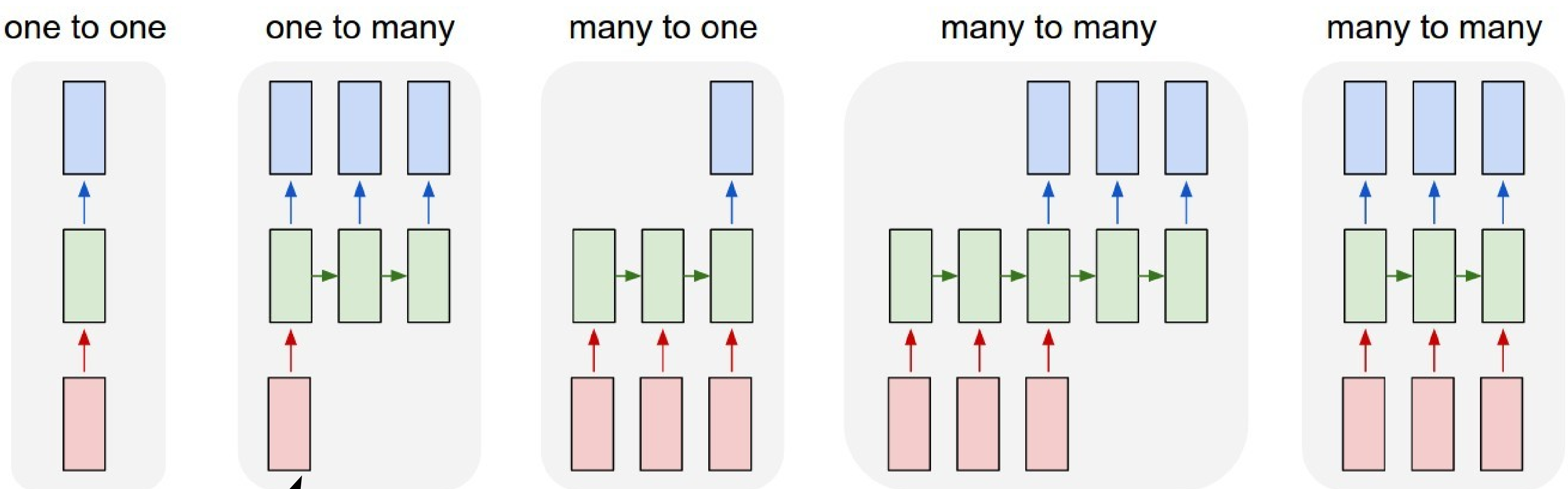


Applications:

- Time series prediction
- Language modeling
- Text sentiment analysis
- Speech recognition
- Translation

http://colah.github.io/

Variations of RNN

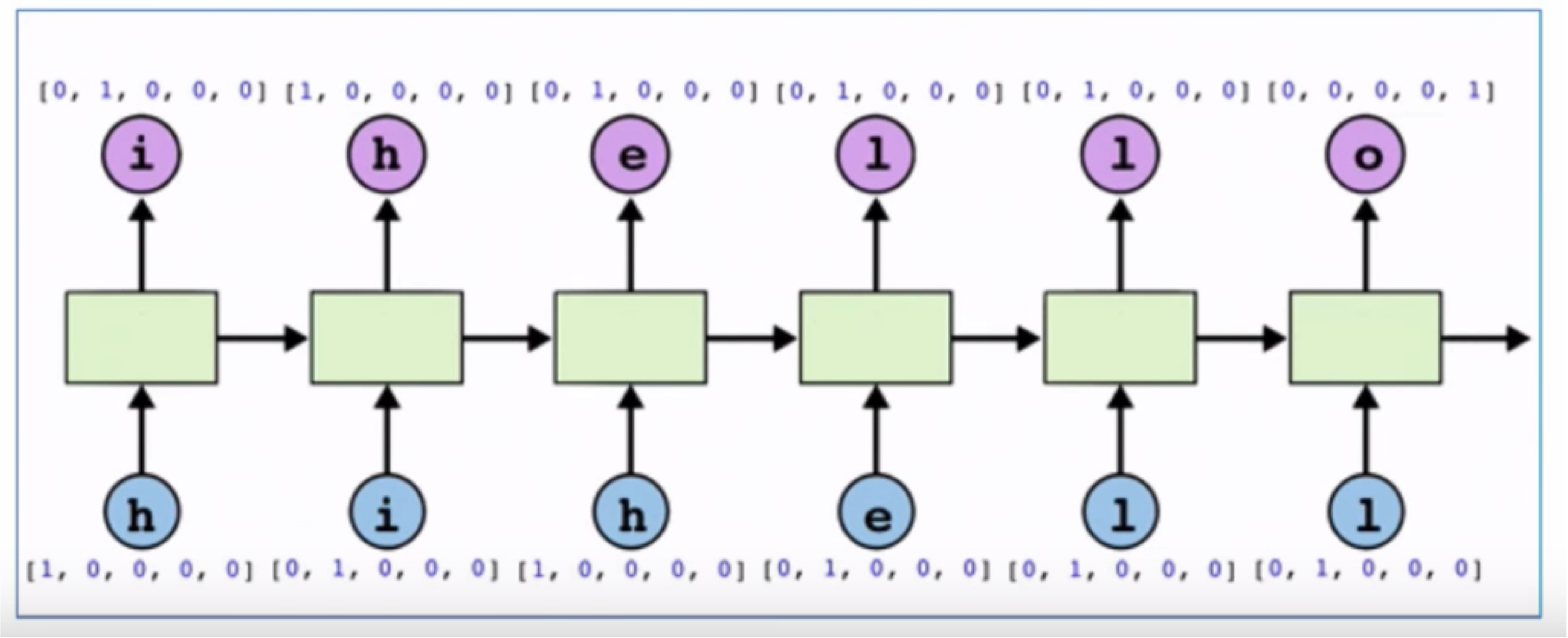


Ex: Image-to-Words

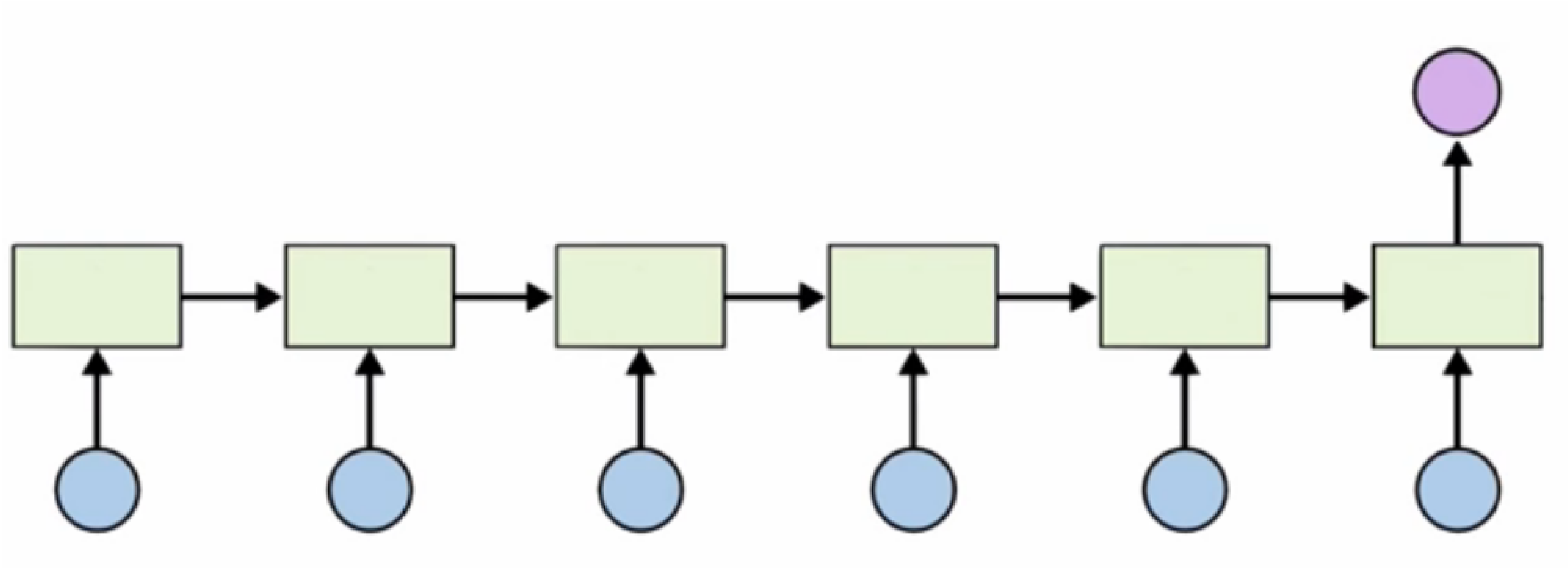
Ex: Sentence-to-Sentiment

Ex: Translation

RNNs- Language modeling

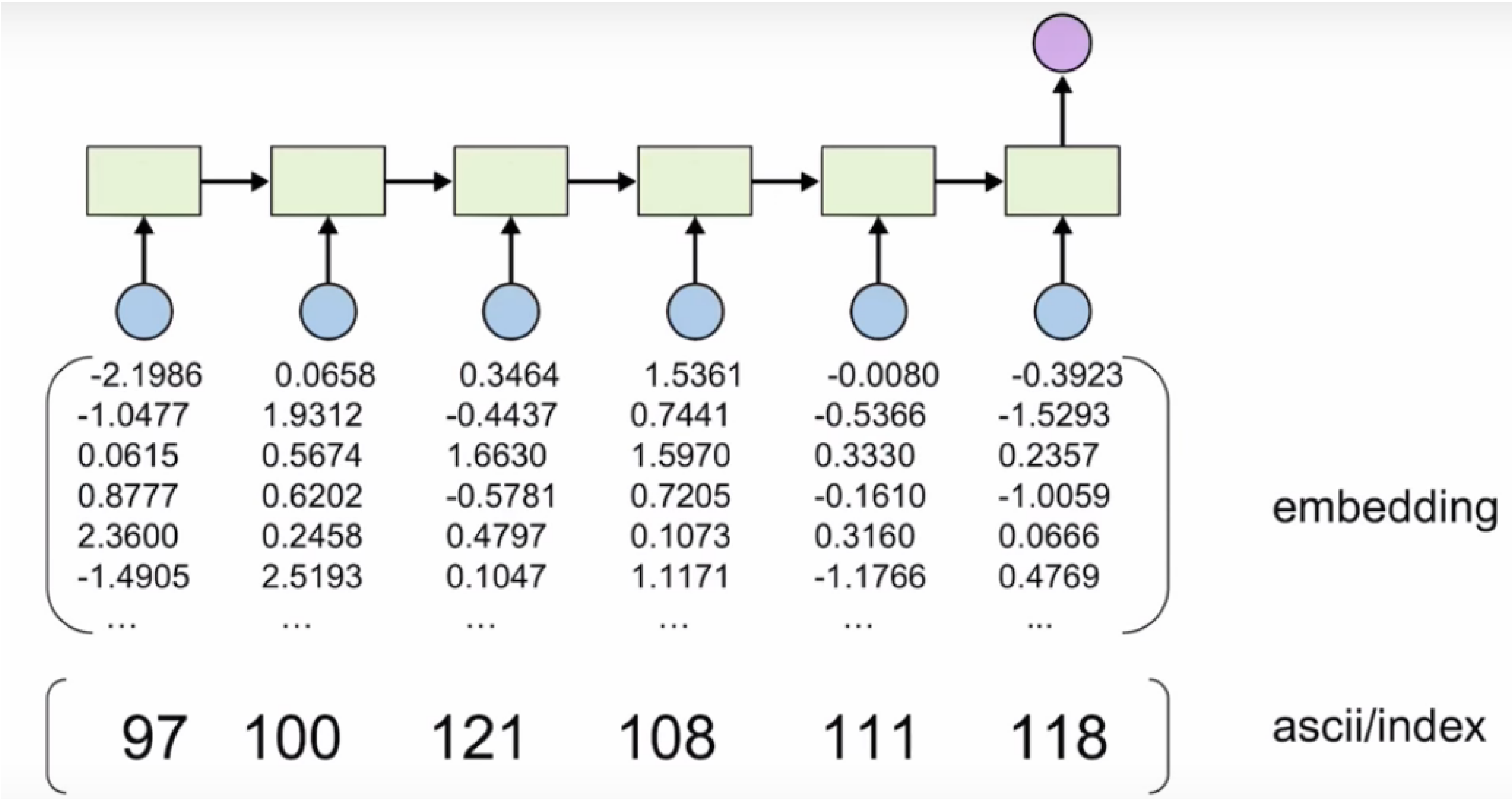


RNNs --Classification



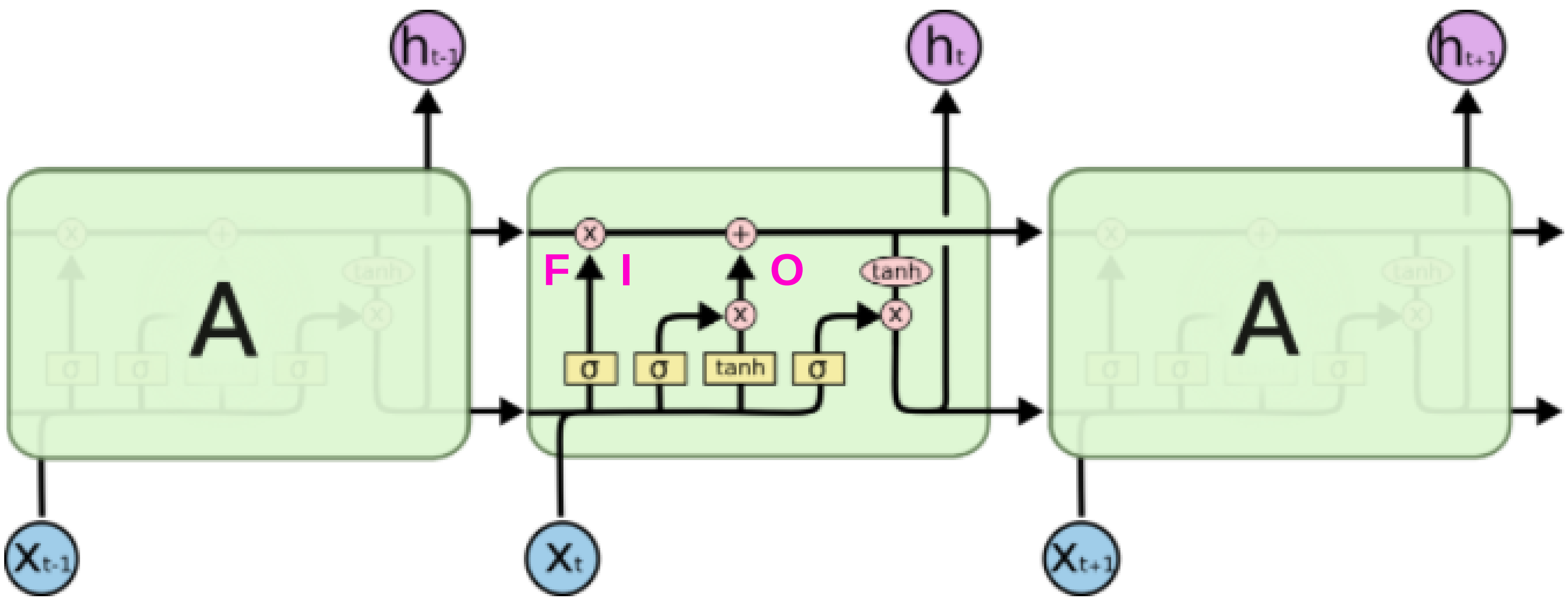
Names	**Language**
• Adalad	German
• Adele	German
• Lin Dan	Chinese
• Chang Lee	Chinese
….	

RNNs--Classification



https://github.com/hunkim

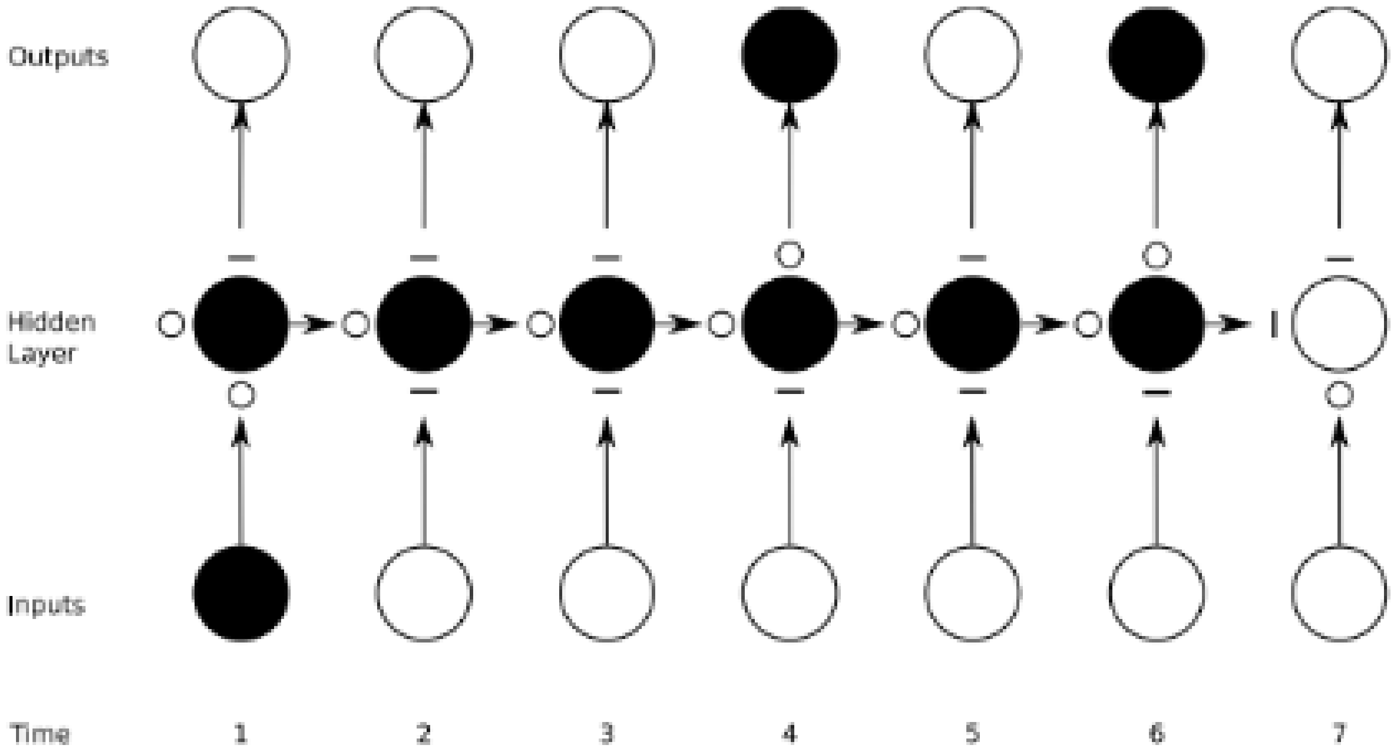
LSTM



Gates:

- Forget gate: Determines whether current contents of memory will be forgotten (erased)
- Input gate: Determines whether the input will be stored in the memory cell
- Output Gate: Determines if current memory contents will be output

LSTM full network



Graves et al 2013

Thank You