Machine Learning for Speech Processing

Dr. Mathew Magimai Doss

October 28, 2022

idiap RESEARCH INSTITUTE EPFL

Outline

Outline

Machine learning

- Learning is an essential part of living
- Learning typically means changing/adapting to be better, as per a given criterion, when a similar situation arrives
- Challenge lies in generalizing to new or unobserved situations

Challenge

- Learning needs training data. We have access to only a finite amount of training data.
- Variabilities in the data
  - Language level: isolated words, sentence, spoken language, read speech, spontaneous speech, dialect ...
  - Speaker level: gender, adult versus child, dialect, age, accent, impaired versus unimpaired (pathological speech), emotion, mood, stress ....
  - Noise
    - Convolutive: recording/transmission condition, reverberation
    - Additive: recording environment, transmission
    - Lombard effect: speaker level variability in noisy environment
- Depending upon the task, a few variabilities are desirable or of interest while others are undesirable or of not interest.

Types of learning

1. Supervised learning
   - Training data is labelled. For example, for a frame of feature vector or a sequence feature vectors we have a "class" label associated to it.
2. Reinforcement learning
   - Training data has partial labels/targets. For example, did a robot carry out the desired action or not?
3. Unsupervised learning
   - Training data does not contain class labels or targets. But, often there is a hidden goal associated with the task. For example, data clustering tasks have a hidden goal such as, minimization of a distance function or maximization of likelihood.

Statistical Pattern Recognition

- Classification
  - Static classification
  - Sequence classification
- Detection: can be regarded as a two class classification problem
- Regression: relation between two variables, namely, measured variable and explanatory variable

Three Key Statistical Rules

1. Bayes's rule:

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

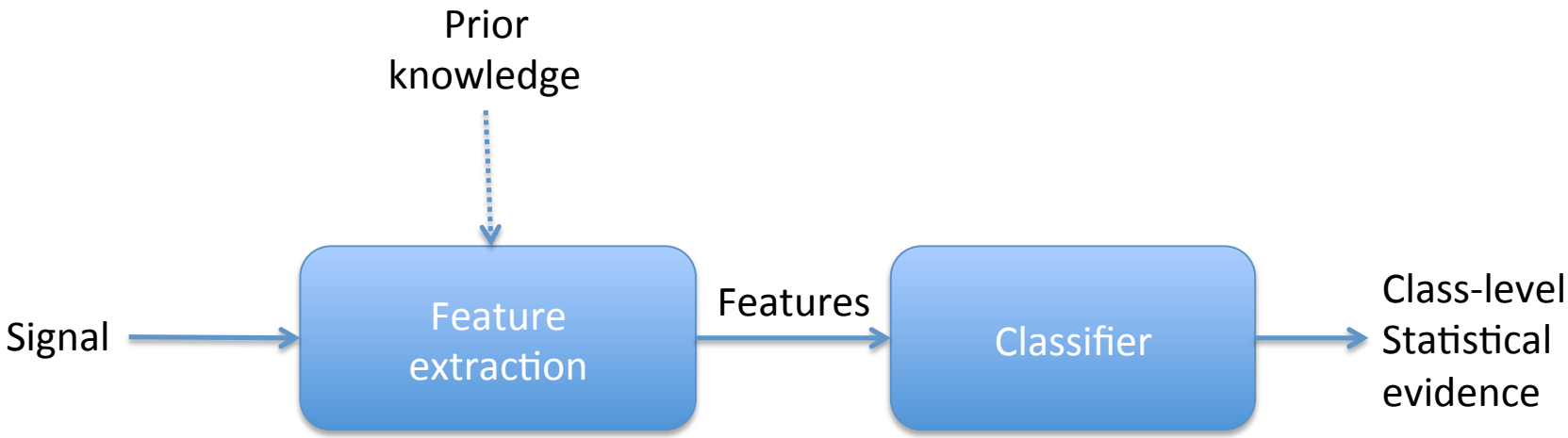
2. If B_k $(k = 1, \ldots, K)$ are mutually exclusive and collectively exhaustive $(\sum_{k=1}^{K} P(B_k) = 1)$

$$P(A) = \sum_{k=1}^{K} P(A, B_k)$$

3. Gibbs sampler:

$$P(B_1, \ldots, B_k, \ldots, B_K) = \prod_{k=1}^{K} P(B_k|B_{k-1}, \ldots, B_1)$$

Static Classification



Speech

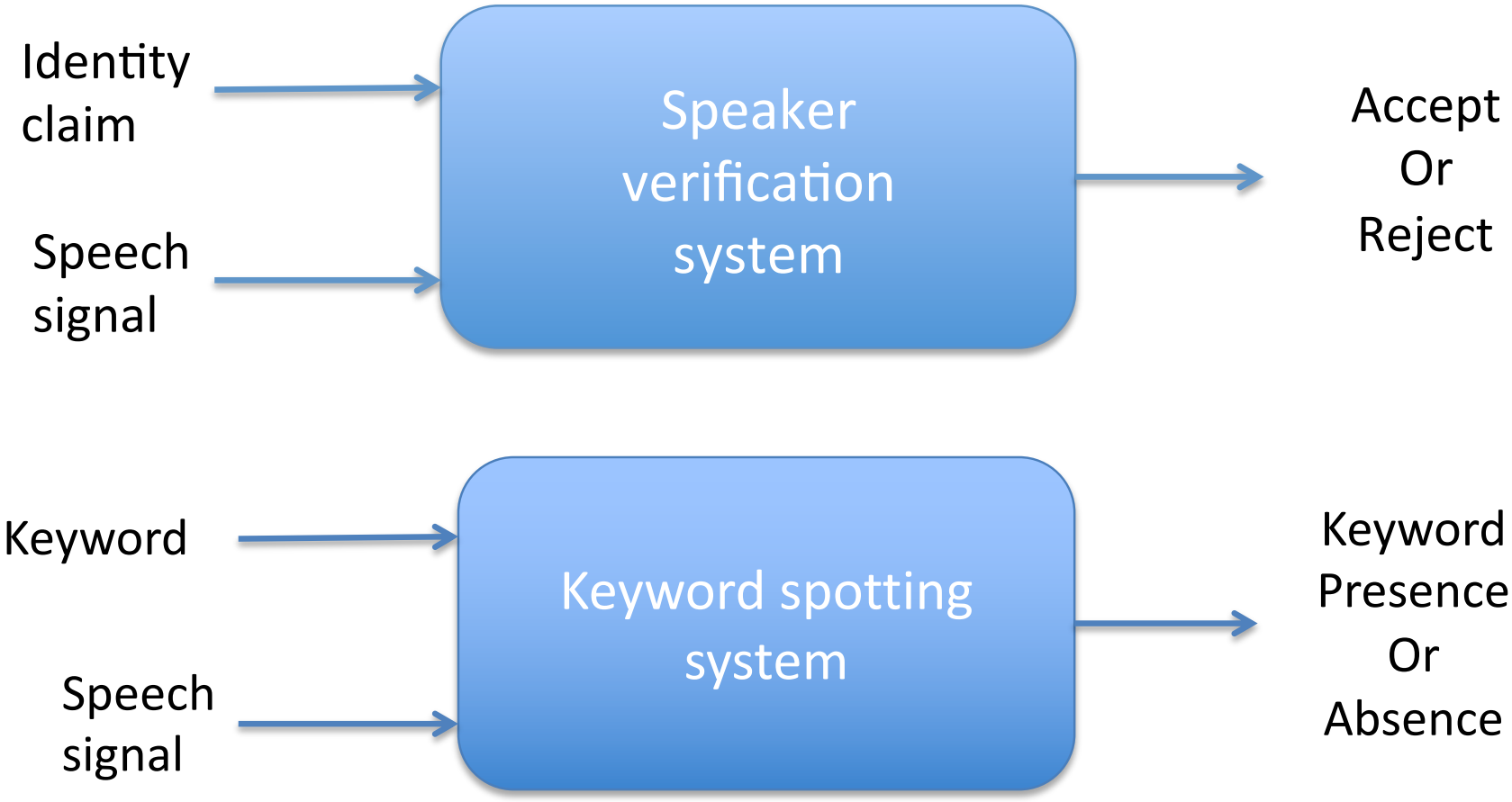
Phoneme
Gender
Speaker
Emotion..

Face image

Person
Age
Gender
Emotion..

Sequence classification

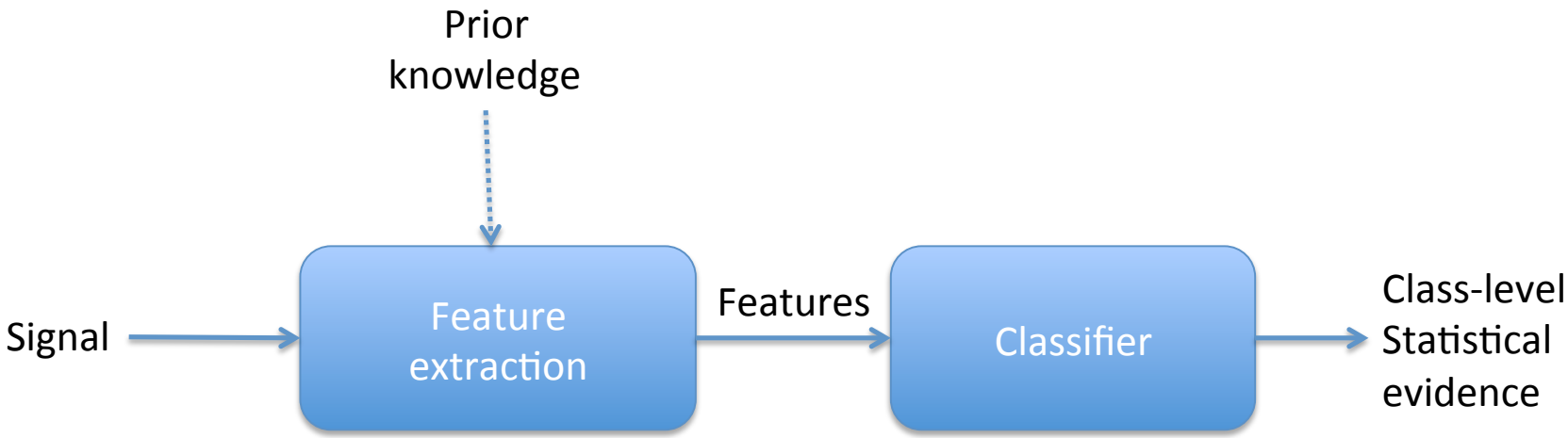
Detection



Hypothesis testing problem

Outline

Static Classification



Speech

Phoneme
Gender
Speaker
Emotion..

Face image

Person
Age
Gender
Emotion..

Theoretical formulation

$$P(C_k|x_m,\Theta) = \frac{p(x_m|C_k,\Theta)\cdot P(C_k|\Theta)}{p(x_m|\Theta)} \ \forall k \in \{1,\cdots K\} \qquad (1)$$

- $P(C_k|x_m,\Theta)$: Posterior probability of class C_k
- $p(x_m|C_k,\Theta)$: Likelihood of class C_k
- $P(C_k|\Theta)$: Prior probability of class C_k
- $p(x_m|\Theta) = \sum_{j=1}^{K} p(x_m|C_j,\Theta)\cdot P(C_j|\Theta)$: Observation likelihood
- Θ: parameters of the statistical model
- $0 \leq P(C_k|x_m,\Theta) \leq 1$ and $\sum_{k=1}^{K} P(C_k|x_m,\Theta) = 1$
- $0 \leq P(C_k|\Theta) \leq 1$ and $\sum_{k=1}^{K} P(C_k|\Theta) = 1$

Generative approach

- Estimate or model $p(x_m|C_k)$ by a probability density function
  - Gaussian or Normal distrution

$$
\begin{aligned}
p(x_m|C_k,\Theta_k) &= N(x_m,\mu_k,\Sigma_k) \\
&= \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{(x_m-\mu_k)^t\Sigma_k^{-1}(x_m-\mu_k)}{2}\right) \\
&\approx \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_k^d} \exp\left(-\frac{1}{2}\left(\frac{x_m^d-\mu_k^d}{\sigma_k^d}\right)^2\right)
\end{aligned}
$$

  - Gaussian mixture models (GMM)

$$
p(x_m|C_k,\Theta_k) = \sum_{j=1}^{J} c_k^j \cdot N(x_m,\mu_k^j,\Sigma_k^j)
$$

- Estimate prior probability $P(C_k)$ (typically done through counting)
- Apply Eqn. (1)
- Θ: means, variance and Gaussian weights (in the case of GMMs)

EM algorithm for GMMs (1)

$$p(x_m) = \sum_{j=1}^{J} c_j p(x_m|\mu_j, \Sigma_j)$$

with $c_j = P(G_j)$ (Weight fot Gaussian j).

Estimation step:

$$P(G_j|x_m) = \frac{c_j p(x_m|\mu_j^{(t)}, \Sigma_j^{(t)})}{\sum_j c_j p(x_m|\mu_j^{(t)}, \Sigma_j^{(t)})}$$

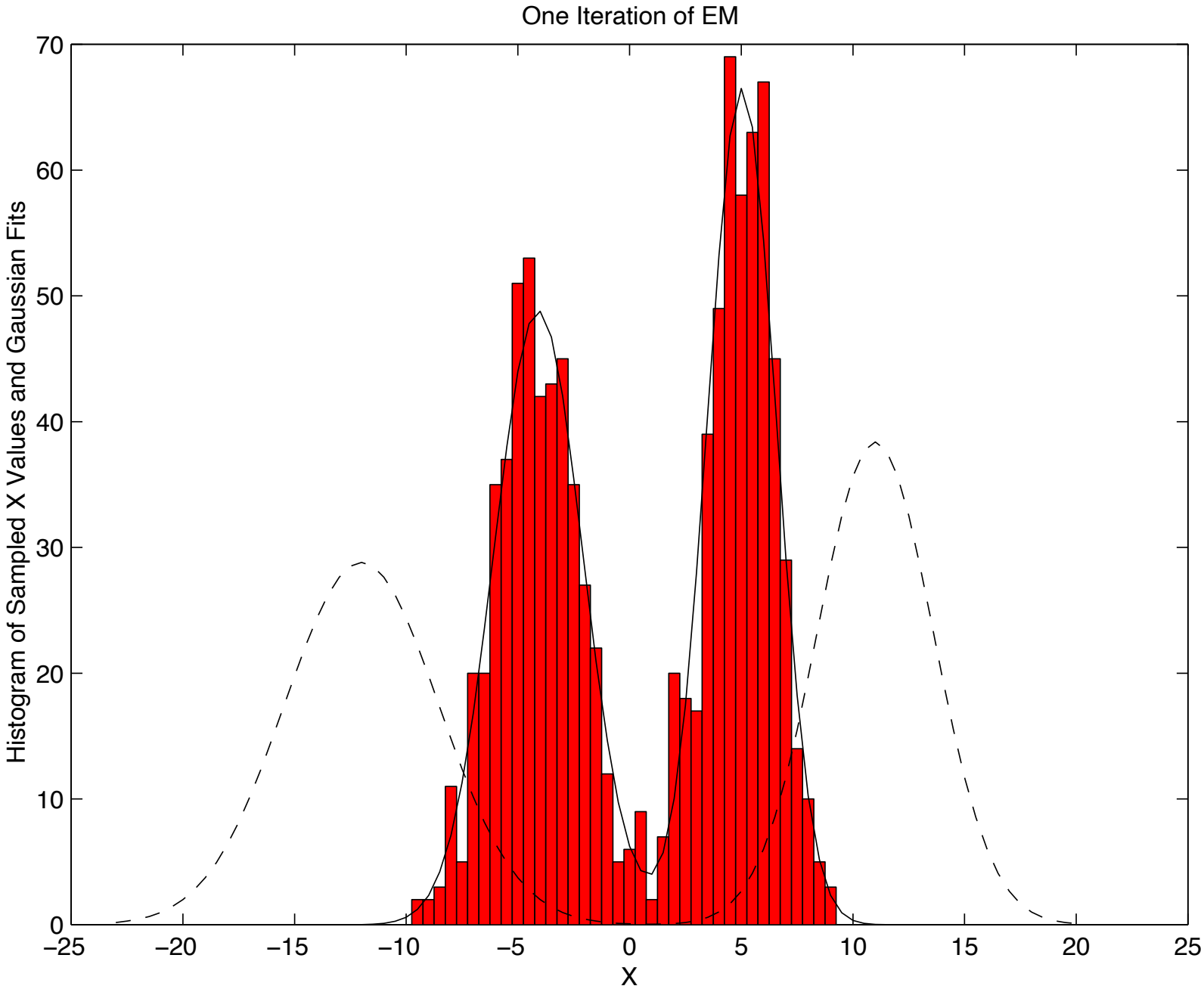
Maximization step:

$$\mu_j^{(t+1)} = \frac{\sum_{m=1}^{n=M} x_m P(G_j|x_m)}{\sum_{m=1}^{m=M} P(G_j|x_m)}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{m=1}^{m=M} P(G_j|x_m)(x_m - \mu_j^{(t+1)})(x_m - \mu_j^{(t+1)})^T}{\sum_{m=1}^{m=M} P(G_j|x_m)}$$

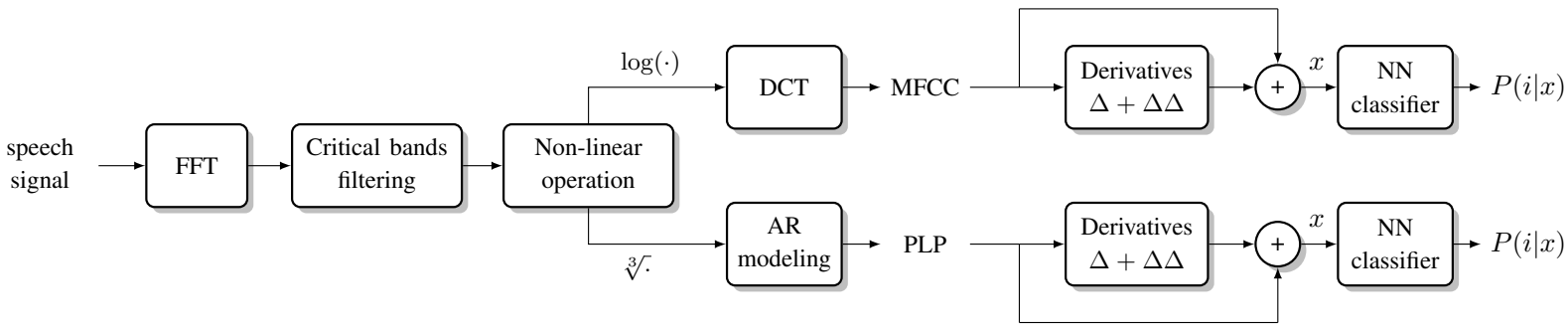
$$c_j^{(t+1)} = \frac{1}{M} \sum_{m=1}^{m=M} P(G_j|x_m)$$

EM algorithm for GMMs (2)



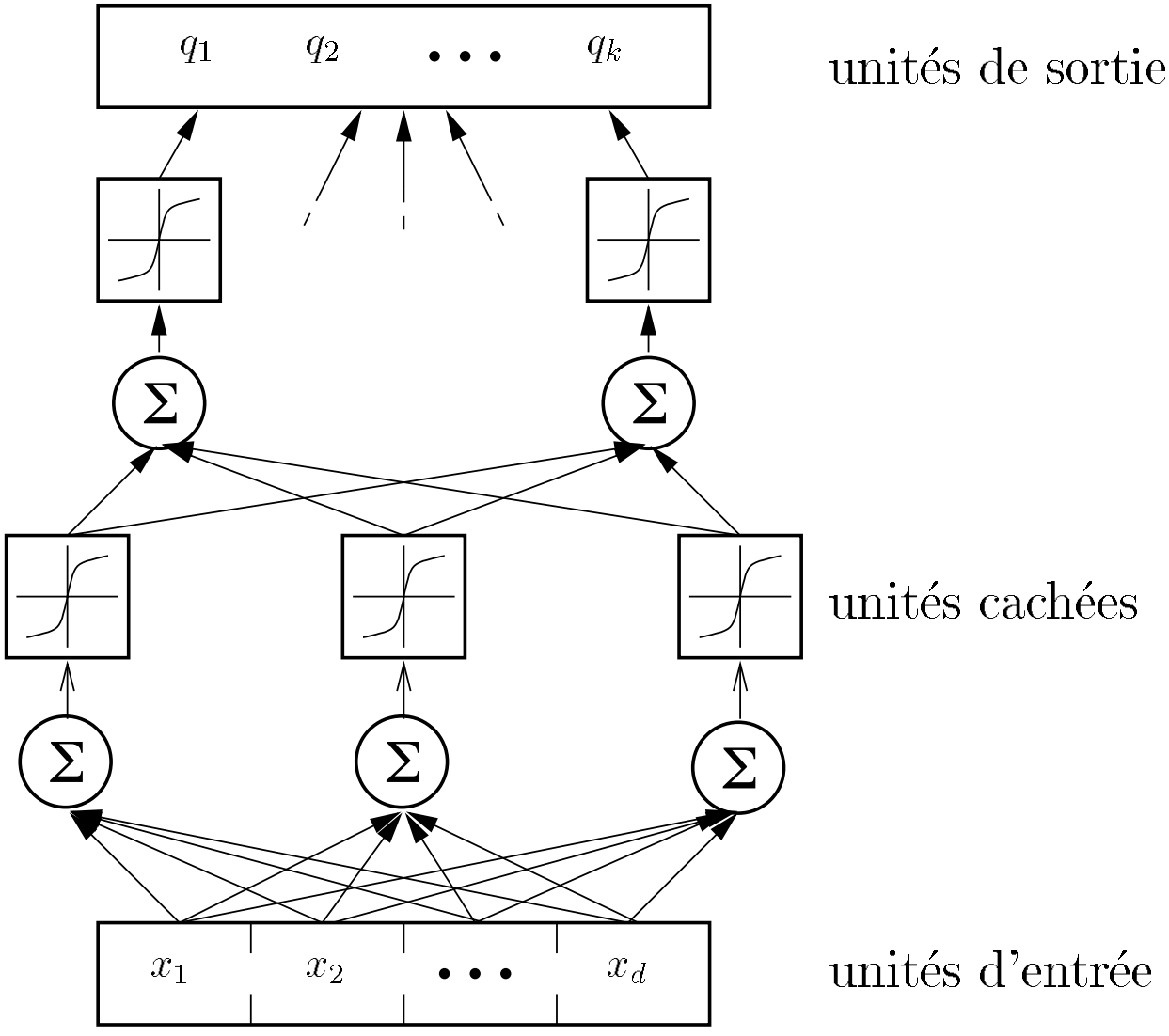
Discriminative approach

- Artificial neural network trained with one hot encoding of target and cross entropy error function (or mean square error function) can directly estimate $P(C_k|x_m, \Theta)$



- Support vector machines (estimation of posterior probability of class not trivial, see Platt's method)
- Θ: parameters of the artificial neural networks (weights and biases) or support vector machines

ANN: Multilayer perceptrons



Typical multilayer perceptron (MLP) architecture, each unit approximating a perceptron.

ANN Training

Supervised training: Input vector sequence:

$$X = \{\mathrm{x}_1, \ldots, \mathrm{x}_n, \ldots, \mathrm{x}_M\}$$

Desired output sequence associated with X:

$$D = \{d(\mathrm{x}_1), \ldots, d(\mathrm{x}_m), \ldots, d(\mathrm{x}_M)\}$$

and $d(\mathrm{x}_m) = (d_1(\mathrm{x}_m), \ldots, d_k(\mathrm{x}_m), \ldots, d_K(\mathrm{x}_m))^{\mathrm{T}}$
In classification mode:

$$d_k(x_m) = \delta_{k\ell} \quad \text{if} \quad \mathrm{x}_m \in C_\ell$$

Parameters Θ: weights and biases

Training Criteria

- Mean Square Error:

$$\underset{\{\Theta\}}{\operatorname{argmin}} \; E = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} [g_k(\mathrm{x}_m, \Theta) - d_k(\mathrm{x}_m)]^2$$

- Entropy or relative entropy:

$$\underset{\{\Theta\}}{\operatorname{argmin}} \; E_e = \sum_{m=1}^{M} \sum_{m=1}^{M} \left[d_k(\mathrm{x}_m) \log \frac{d_k(\mathrm{x}_m)}{g_k(\mathrm{x}_m, \Theta)} + (1 - d_k(\mathrm{x}_m) \log \frac{1 - d_k(\mathrm{x}_m)}{1 - g_k(\mathrm{x}_m, \Theta)} \right]$$

$g_k(\mathrm{x}_m, \Theta)$ denotes the output of the neural network.

Error back propagation training

Minimization of E (or E_e) in the parameter space Θ (weights + biases)

$$\frac{\bigtriangledown E}{\bigtriangledown \Theta} = 0$$

Done via a gradient procedure:

$$\bigtriangleup w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

α denotes learning rate

Adjust w_{ij} based on $\bigtriangleup w_{ij}$.

Offline Error Back-Propagation

Initialize network at random; choose "large" learning rate
Until convergence = true
    For m=1 to M
        Forward computation of $g_k(x_m, \Theta)$
        Error calculation and global error update
        Error backward propagation
        and compute local $\delta\Theta(x_m)$
    Θ **update** $= \sum_{m=1}^{M} \delta\Theta(x_m)$
    If error (on cross-validation set) decreases
        save new parameters
    Otherwise, don't save new parameters
        and decrease learning rate
If learning rate $<$ threshold then convergence = true;

Online Error Back-Propagation

Initialize network at random
Choose "large" learning rate; convergence = false
Until convergence = true
    For m=1 to M (or something else)
        Pick x_m at random
        Forward computation of $g_k(x_m, \Theta)$
        Error calculation
        Error backward propagation and Θ **update**
    If error (on cross-validation set) decreases
        save new parameters
    Otherwise, don't save new parameters
        and decrease learning rate
If learning rate $<$ threshold: convergence = true;

In practice: Mini-batch training, a combination of offline and online error back propagation.

Cross-validation training (1)

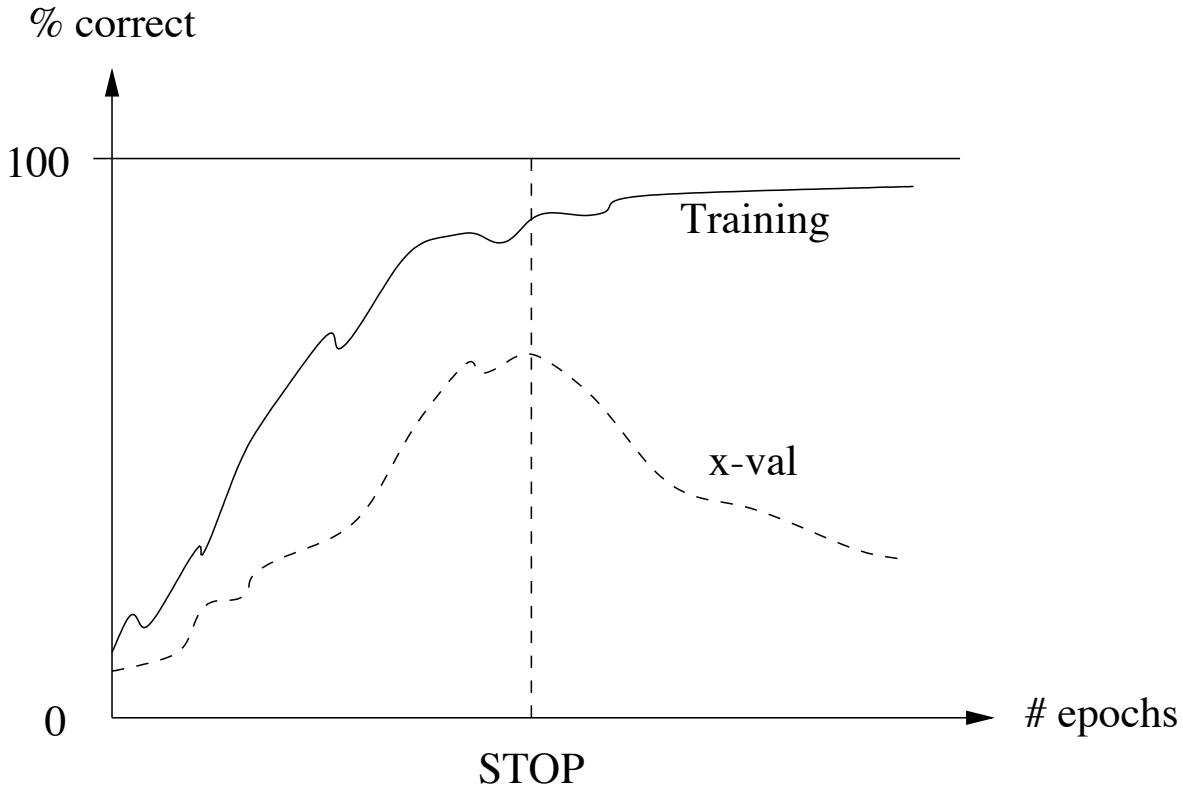
After each MLP training epoch:

1. check recognition performance on independent data set
2. stop training if rec performance starts to decrease and learning rate below a given threshold

Remarks: there are other solutions like

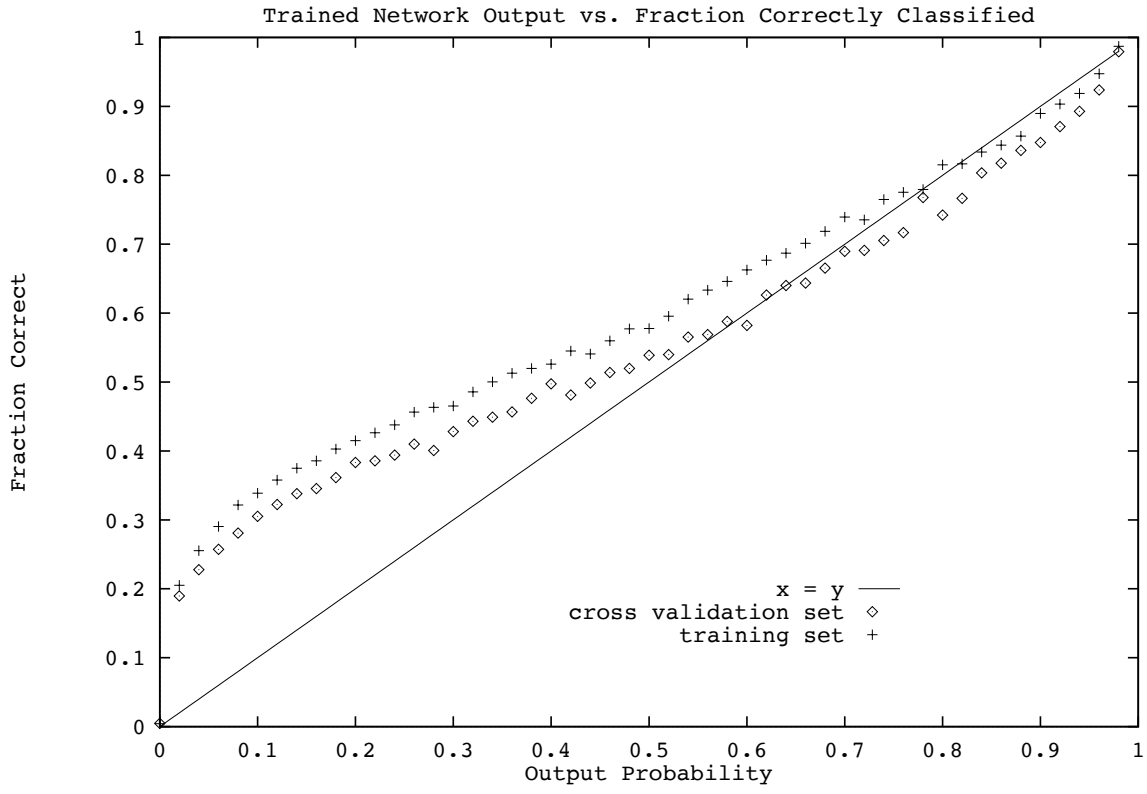
- Forcing small weights
- "Optimal Brain Damage"
- Regularized training (Bayesian approach)

Cross-validation training (2)



Example of crossvalidation training. x-val represents the crossvalidation data on which classification performance is regularly checked. Training is stopped when performance on x-val data reaches the maximum.

Interpretation of ANN output $g_k(\mathrm{x}_m, \Theta)$



ANN output $g_k(\mathrm{x}_m, \Theta)$ is an estimate of posterior probability of class C_k, i.e. $P(C_k|\mathrm{x}_m)$.

Decision making

- Maximum likelihood

$$C^*_{mle} = \arg\max_k p(X|C_k, \Theta)$$

$$p(X|C_k, \Theta) = \prod_{m=1}^{M} P(x_m|C_k, \Theta) (\text{Assuming i.i.d})$$

- Maximum aposteriori probability

$$C^*_{map} = \arg\max_k p(C_k|X, \Theta)$$

$$P(C_k|X, \Theta) = \frac{1}{Z} \cdot \prod_{m=1}^{M} P(C_k|x_m, \Theta) (\text{Assuming i.i.d})$$

or

$$P(C_k|X, \Theta) = \frac{1}{Z} \cdot \sum_{m=1}^{M} P(C_k|x_m, \Theta)$$

Z is a normalization factor.

Better to perform computation using logarithm to avoid underflow issues.

Outline

Sequence classification

Statistical ASR

Given feature observation $X = \{x_1, \cdots x_m, \cdots x_M\}$ predict the most probable word sequence $W^* = \{w_1^* \cdots w_n^* \cdots w_N^*\}$

$$\begin{aligned} W^* &= \arg\max_{W_i \in \mathcal{W}} P(W_i|X, \Theta) \\ &= \arg\max_{W_i \in \mathcal{W}} \frac{p(X|W_i, \Theta_a) \cdot P(W_i|\Theta_l)}{p(X|\Theta)} \\ &= \arg\max_{W_i \in \mathcal{W}} p(X|W_i, \Theta_a) \cdot P(W_i|\Theta_l), \end{aligned}$$

where W_i denotes a word hypothesis, $\mathcal{W}$ denotes a set of word hypotheses and $\Theta = \{\Theta_a, \Theta_l\}$

- Acoustic modeling: estimation of $p(X|W_i, \Theta_a)$ using hidden Markov models (HMMs)
- Language modeling: estimation of $P(W_i|\Theta_l)$ using discrete Markov models (DMMs)

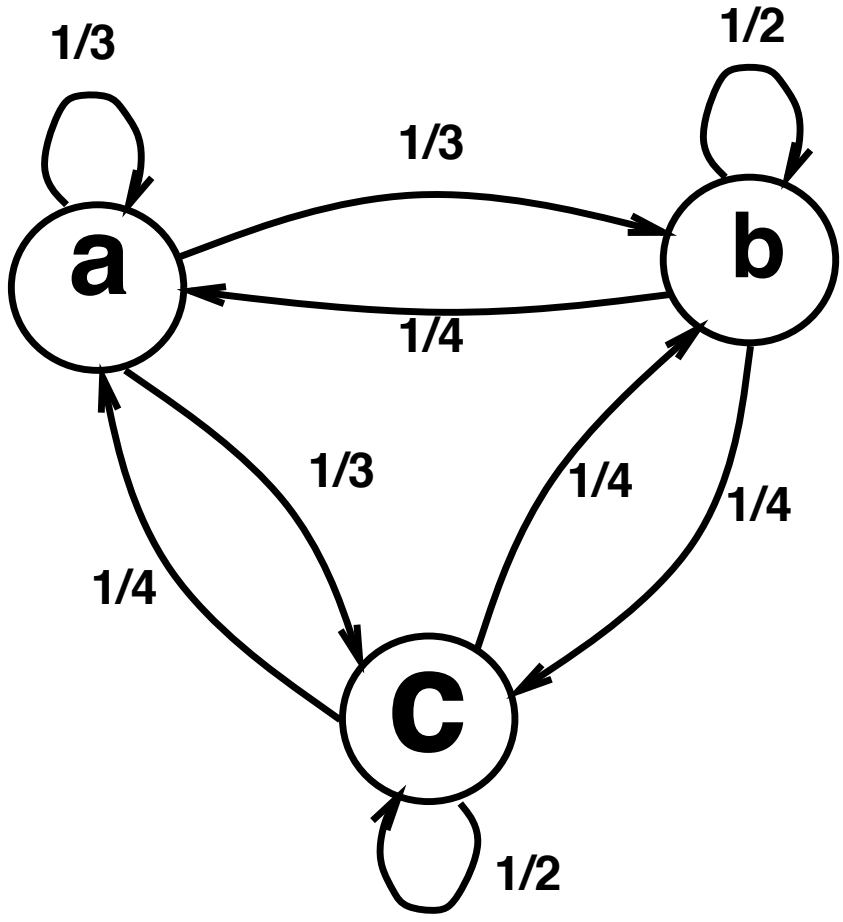
Discrete Markov Model (DMM)

- Stochastic finite state automaton
- *MM* built up from states q_ℓ from a set of classes (states) $\Omega = \{\omega_1, \ldots, \omega_k, \ldots, \omega_K\}$
- q^n particular state of M visited at time n,
- $q_\ell^n \equiv \{q^n = q_\ell\}$, $q_l \in \Omega$
- *MM* is defined by topology, i.e., how the states are connected
  See HMM lab exercise 1
- Parametrized by:

$$\begin{aligned} P(q_\ell^n | q_k^{n-1}, q_j^{n-2}, \ldots) &\simeq P(q_\ell^n | q_k^{n-1}) \quad (\textit{1st order Markov}) \\ &\simeq P(q_\ell | q_k) = P_{k\ell} \quad (\textit{time independent}) \end{aligned}$$

Note: $q_l, q_k \in \Omega$
Transition probability matrix: $A = \{P_{k,\ell}\}$.

DMM (2)



Example of fully connected discrete Markov model with $\Omega = \{a, b, c\}$. For example, in case of weather model: "a" = "cloudy", "b"= "rainy" and "c" = "sunny"

Typical Problems (1)

- **Probability of a particular path**

$$\begin{aligned} P(Q|MM) &= P(q^1|q_i^0)P(q^2|q^1)\dots P(q^n|q^{n-1})\dots P(q^N|q^{N-1}) \\ &= \prod_{n=1}^{N} P(q^n|q^{n-1}) \end{aligned}$$

One way to estimate $P(W_i|\Theta_l)$, e.g.

$$P(W^*|\Theta_l) = \prod_{n=1}^{N} P(w_n^*|w_{n-1}^*)$$

- **State duration distribution**

Probability to stay in state q_i for *exactly* d time steps?

$$Q = \{q_i^0, q_i^1, q_i^2, \dots, q_i^d, q_j^{d+1}\}, \text{ with } j \neq i$$

and:

$$P(Q|MM) = (P_{ii})^{d-1}(1 - P_{ii})$$

See HMM lab exercise 1

Typical Problems (2)

- **Probability to go from state q_i to q_j in N steps**

$$P(q_j^N|q_i^0) = \sum_{n=0}^{N} \sum_{\ell=1}^{L} P(q_j^N, q_\ell^n|q_i^0)$$

Defining

$$\alpha(\ell, n) = P(q_\ell^n|q_i^0, N)$$

We have:

$$\alpha(\ell, n+1) = \sum_k \alpha(k, n) P_{k\ell}$$

$$P(q_j^N|q_i^0) = \alpha(j, N)$$

Typical Problems (3)

- **Probability of best path of length N between q_i and q_j**
  If $\overline{P}(k, n)$ is probability of best path to go from q_i to q_k in n steps:

$$\overline{P}(\ell, n+1) = \max_k \overline{P}(k, n) P_{k\ell}$$

  and

$$\overline{P}(q_j^N | q_i^0) = P(j, N)$$

  Generalization:

$$A^n(i, j) = P(q_j^n | q_i^0)$$

DMM training

- Training data

  Seq 1: *R, R, R, S, S, C, R, C, …….*

  Seq 2: *S, C, S, C, C, R, ……*

  .

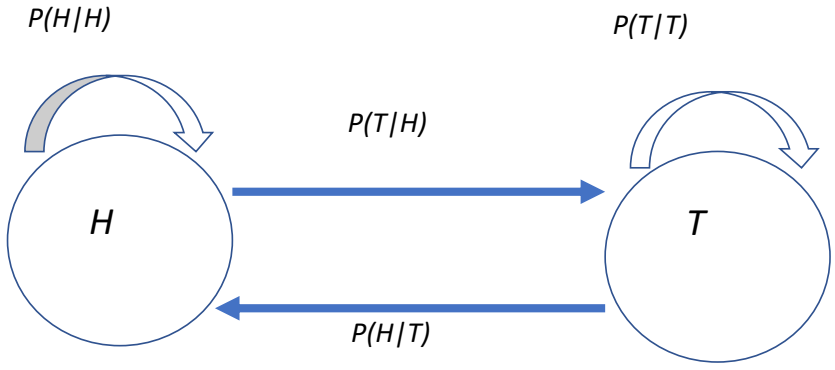
  .

  Seq L: *S, R, C, R, C, R, C, S, S, ……*
- *P(S|I)* = Count(seq starting with S)/L, *P(R|I)* = Count(seq starting with R)/L ….. (L denotes number of sequences in the training data)
- *P(S|R)* = Count(R->S)/Count(R), *P(C|S)* = Count(S->C)/Count(S), *P(R|R)* = Count(R->R)/Count(R) ……

Another approach: train a neural network, e.g., recurrent neural network

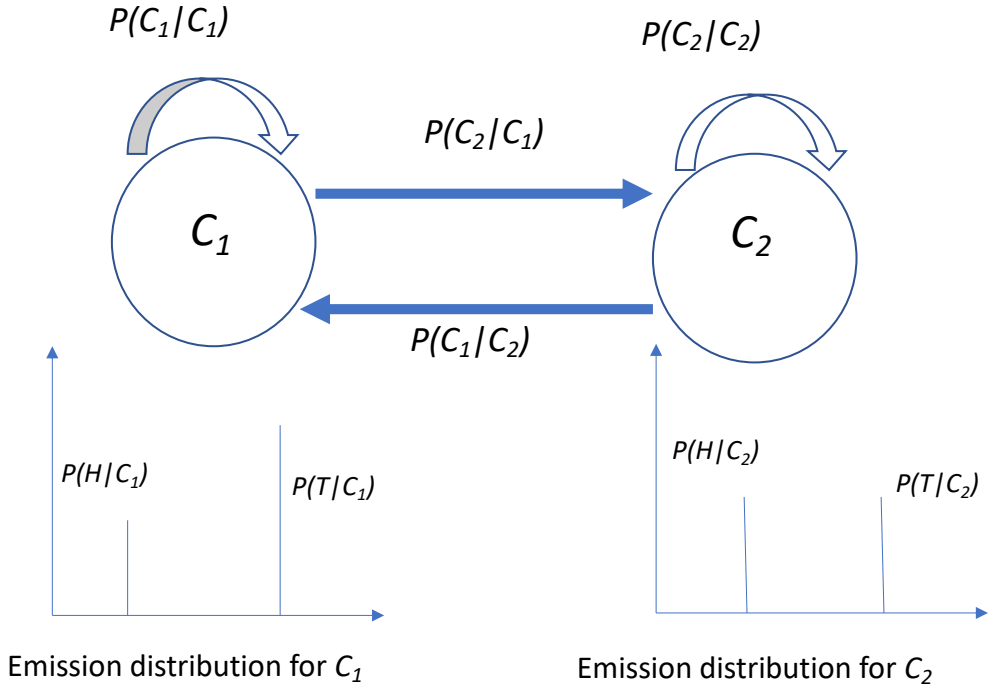
From DMM to HMM (1)

- Coin tossing and only the results of each coin toss i.e. Heads (H) or Tails (T) is revealed
  $H\ H\ T\ T\ T\ H\ H\ H\ H\ T\ T \ldots$
- 1-coin model: DMM with two states $\Omega = \{H, T\}$ and all transition probabilities are equal to 0.5.



From DMM to HMM (2)

- 2-coin model: HMM with two states $\Omega = \{C_1, C_2\}$; each state associated with $P(H|C_i)$ and $P(T|C_i)$ $i \in \{1, 2\}$; and the transition probability reflects the probability of choosing one of the (possibly biased) coins



Estimation of $P(X|W_i, \Theta_a)$ using HMMs

$$\begin{aligned} P(X|W_i, \Theta_a) &= \sum_{Q \in W_i} P(X, Q|W_i, \Theta_a) \\ &= \sum_{Q \in W_i} P(X|Q, W_i) P(Q|W_i), \end{aligned}$$

where $Q = \{q_1, \cdots q_m, \cdots q_M\}$ denotes sequence of HMM states.
After i.i.d and first order Markov assumption

$$\begin{aligned} P(X|W_i, \Theta_a) &= \sum_{Q \in W_i} \prod_{m=1}^{M} p(x_m|q_m) \prod_{m=1}^{M} P(q_m|q_{m-1}) \\ &= \sum_{Q \in W_i} \prod_{m=1}^{M} p(x_m|q_m) P(q_m|q_{m-1}) \text{ Full likelihood} \\ &\approx \max_{Q \in W_i} \prod_{m=1}^{M} p(x_m|q_m) P(q_m|q_{m-1}) \text{ Viterbi approx.} \end{aligned}$$

For the sake of simplicity, Θ_a and W_i are dropped in the latter equations.
$p(x_m|q_m)$ is referred to as emission likelihood and $P(q_m|q_{m-1})$ state transition probabilities. *See HMM lab exercises 3 and 4.*

Estimation of emission likelihood

- Gaussians and Gaussian-Mixtures

$$
\begin{aligned}
p(x_m|q_m = k, \Theta_a) &= N(x_m, \mu_k, \Sigma_k) \\
&= \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{(x_m - \mu_k)^t \Sigma_k^{-1}(x_m - \mu_k)}{2}\right)
\end{aligned}
$$

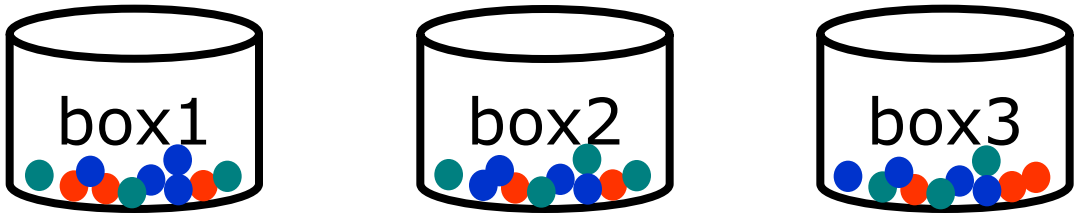
$$\text{OR}:$$

$$
p(x_m|q_m = k, \Theta_a) = \sum_{j=1}^{J} c_k^j N(x_m, \mu_k^j, \Sigma_k^j)
$$

- Artificial neural networks: estimate scaled-likelihood $p_{sl}(x_m|q_m = k, \Theta_a)$

$$
p_{sl}(x_m|q_m = k, \Theta_a) = \frac{p(x_m|q_m = k, \Theta_a)}{p(x_m|\Theta_a)} = \frac{P(q_m = k|x_m, \Theta_a)}{P(q_m = k|\Theta_a)}
$$

Hidden Markov Model (HMM)



P(r|B1)
P(b|B1)
P(g|B1)

P(r|B3)
P(b|B3)
P(g|B3)

P(r|B3)
P(b|B3)
P(g|B3)

} **Emission probabilities**

P(B1|B1), P(B2|B1), P(B3|B1)
P(B1|B2), P(B2|B2), P(B3|B2)
P(B1|B3), P(B2|B3), P(B3|B3)

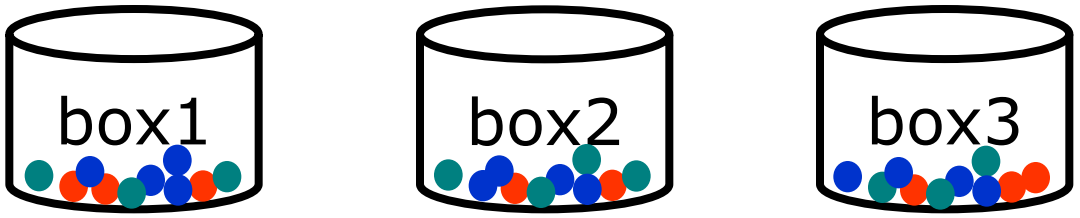
} **Transition probabilities**

Double, embedded stochastic process:

-choose box using transition probs.

-choose ball using emission probs.

Hidden Markov Model (HMM)



$$P(r,b,r|Model) = P(r,b,r|B1,B1,B1)\,P(B1,B1,B1) + P(r,b,r|B1,B1,B2)\,P(B1,B1,B2) + P(r,b,r|B1,B1,B3)\,P(B1,B1,B3) + P(r,b,r|B1,B2,B1)\,P(B1,B2,B1) + P(r,b,r|B1,B2,B2)\,P(B1,B2,B2) + \ldots + P(r,b,r|B3,B3,B3)\,P(B3,B3,B3)$$

with $P(r,b,r|Bi,Bj,Bk) = P(r|Bi)\,P(b|Bj)\,P(r|Bk)$

Conditional i.i.d assumption, i.e., given the state at any time instance the observation at that time instance/frame is only dependent on that state and independent of all observations and states at other time instances/frames

$$P(Bi,Bj,Bk) = P(Bi)\,P(Bj|Bi)\,P(Bk|Bj)$$

First order Markov assumption

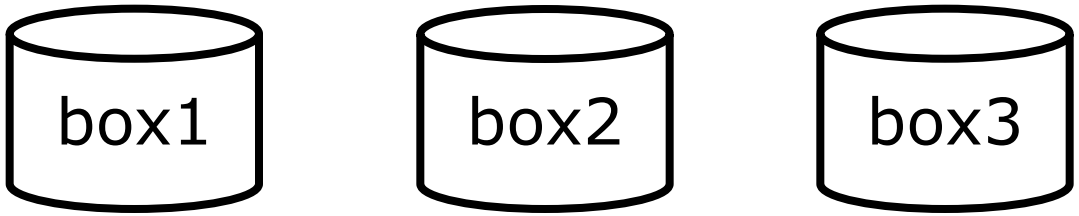
Hidden Markov Model (HMM)

- **Estimation** problem : $P(\boldsymbol{X}|\boldsymbol{M}_w)$?
  O(Nstates^{Nobs}) computations???

  Dynamic programming

  Baum-Welch (all paths) ***Viterbi*** (best path only)
- **Training** problem: best emission and transition probs, given model topology and data
- **Decoding** problem: best sequence of states
  ➔ ***Viterbi*** again

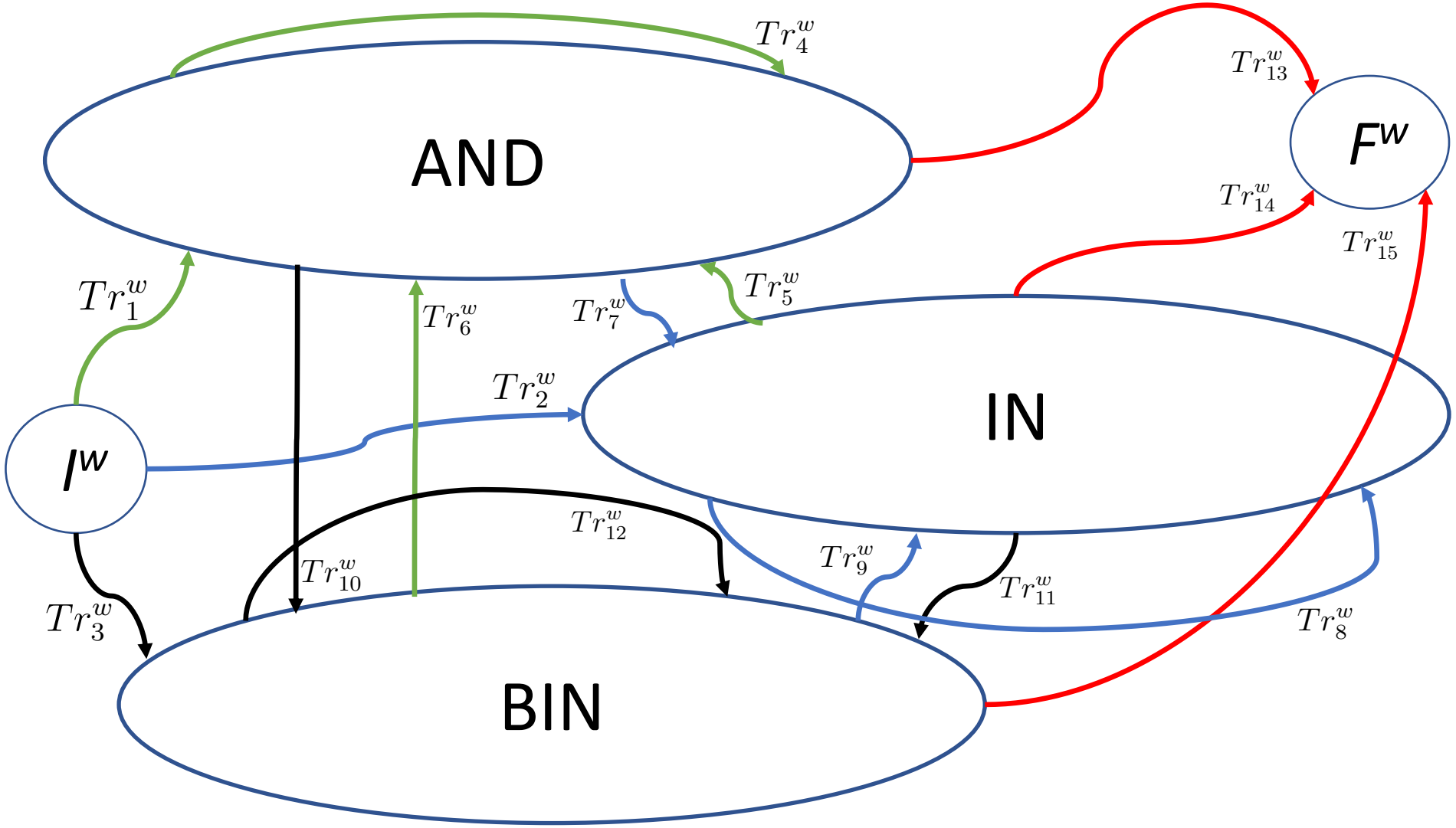
Training HMMs

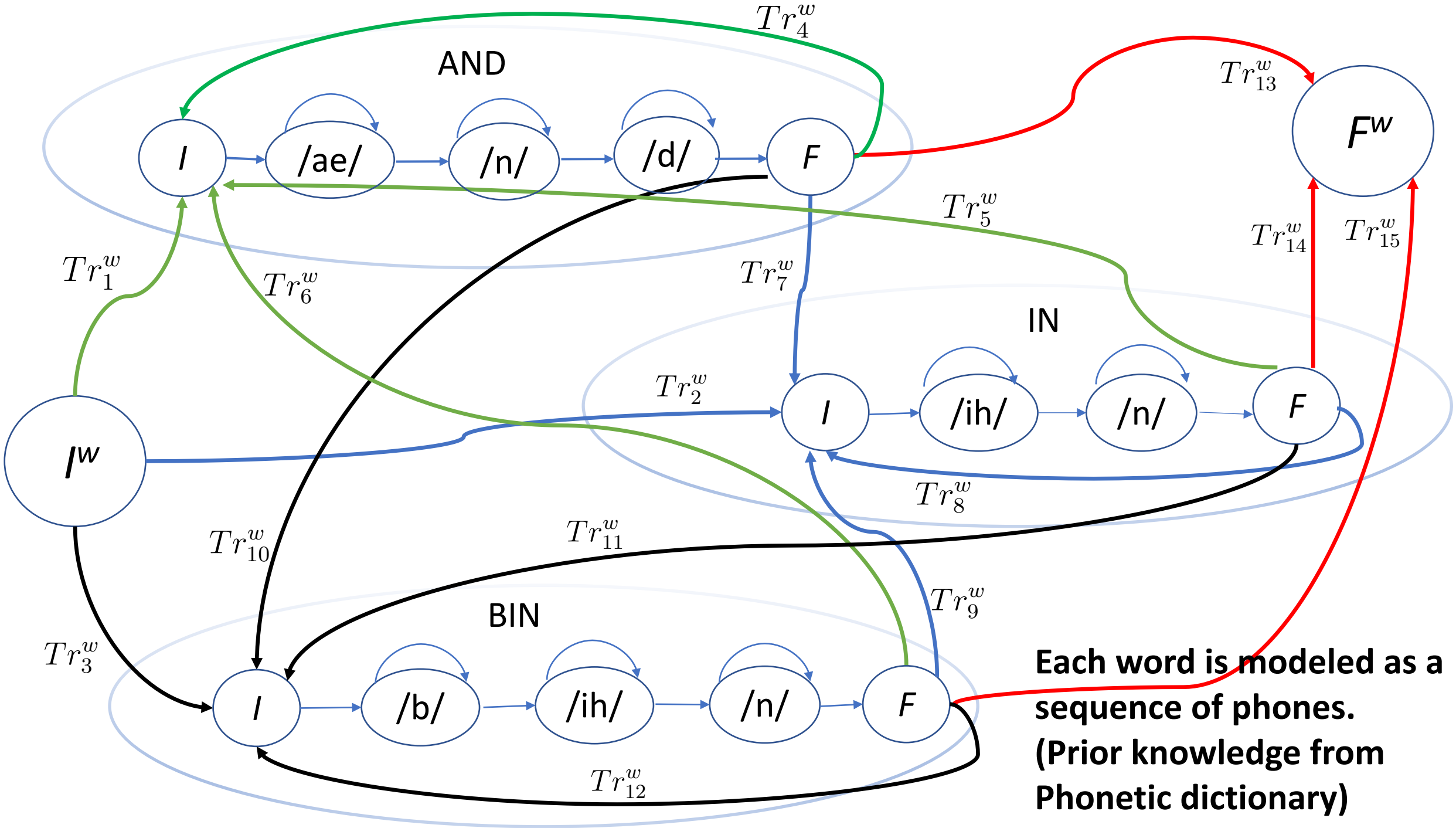


- Data:« brbrrbgbrbbgbggbrbgbrggbrbgg… »
- Emission and transition probabilities?
  If states were known: counting
- **EM** (expectation-maximization) **algorithm**
  - Initialize Probs. (first guess if possible): $\mathbf{M_w^0}$
  - Decode the data with $\mathbf{M_w^0}$ → states
  - Re-estimate Probs. by counting: $\mathbf{M_w^1}$

  until $\mathbf{M_w^j \approx M_w^{j+1}}$

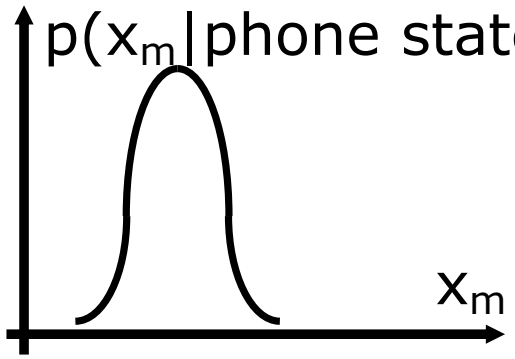
Word sequence modeling (DMM)



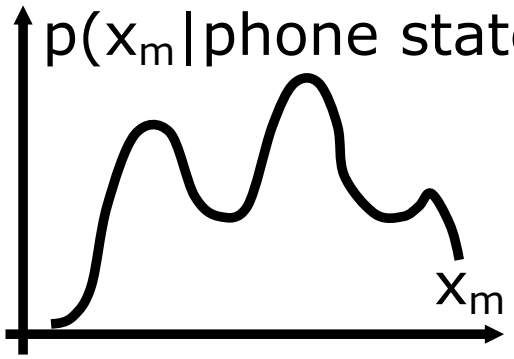

AND
I
/ae/
/n/
/d/
F
IN
I
/ih/
/n/
F
BIN
I
/b/
/ih/
/n/
F
I^w
F^w
Tr_1^w
Tr_2^w
Tr_3^w
Tr_4^w
Tr_5^w
Tr_6^w
Tr_7^w
Tr_8^w
Tr_9^w
Tr_{10}^w
Tr_{11}^w
Tr_{12}^w
Tr_{13}^w
Tr_{14}^w
Tr_{15}^w
Each word is modeled as a sequence of phones. (Prior knowledge from Phonetic dictionary)

Each phone is modeled by an HMM

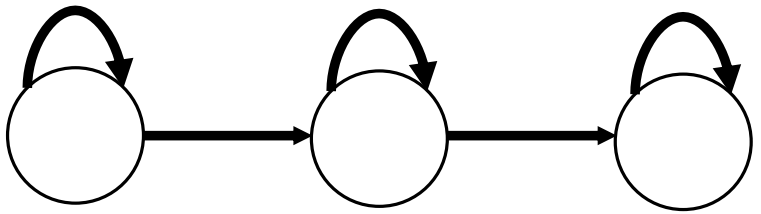
- Observation= x_m ⇨ $p(x_m|phone\ state)$=***continuous***
  Cannot be estimated by counting
  Estimated via the p.d.f. of a distribution
  - ex: Gaussian

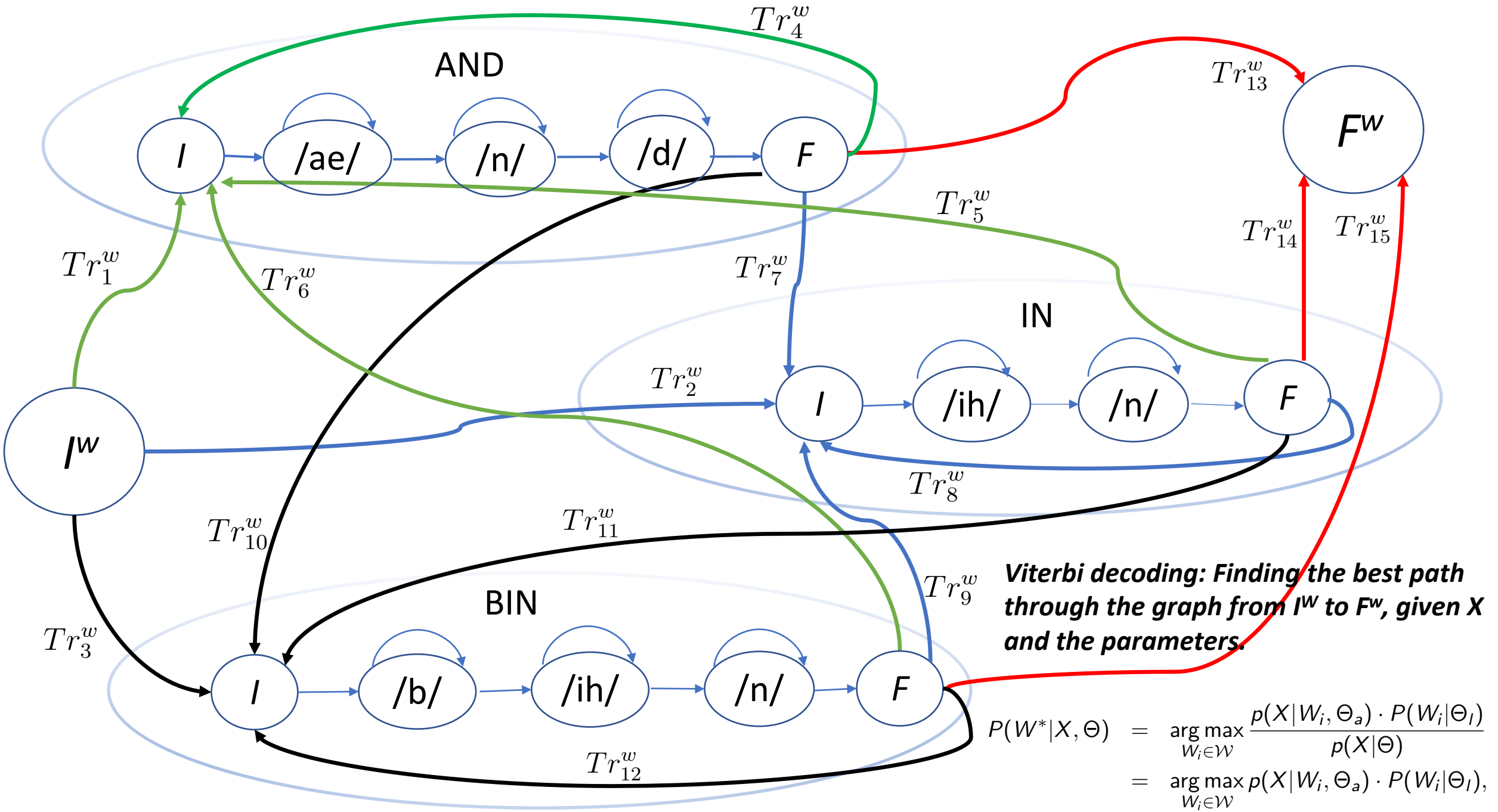


Multi-Gaussian



- In practice, each phoneme is modeled as 3 states (*Bakis* model)





Viterbi decoding: Finding the best path through the graph from I^w to F^w, given X and the parameters.

$$P(W^*|X,\Theta) = \arg\max_{W_i \in \mathcal{W}} \frac{p(X|W_i,\Theta_a) \cdot P(W_i|\Theta_l)}{p(X|\Theta)}$$
$$= \arg\max_{W_i \in \mathcal{W}} p(X|W_i,\Theta_a) \cdot P(W_i|\Theta_l),$$

Outline

Detection

$$\frac{p(X|\text{genuine speaker})}{p(X|\text{impostor})} \geq \delta_{asv}$$

Challenge: modeling or estimation of $p(X|\text{impostor})$

$$\frac{p(X|\text{keyword})}{p(X|\text{not keyword})} \geq \delta_{kws}$$

Challenge: modeling or estimation of $p(X|\text{not keyword})$

Two types of error: False negative, False positive

Thank you for your attention!

Dr. Mathew Magimai Doss

Idiap Research Institute, Martigny, Switzerland