(Deep) Neural Networks for Speech Processing

Kate Knill

September 2015

Cambridge University Engineering Department

DREAMS Summer School Tutorial 2015

Overview

- Part 1:
  - Motivation
  - Basics of Neural Networks
  - Voice Activity Detection
  - Automatic Speech Recognition
- Part 2:
  - Neural Networks for ASR Features and Acoustic Models
  - Neural Networks for Language Modelling
  - Other Neural Network Architectures

Motivation

Speech processing sequence-to-sequence mapping tasks

Speech (continuous time series) $\rightarrow$ Speech (continuous time series)

– Speech Enhancement, Voice Conversion

Speech (continuous time series) $\rightarrow$ Text (discrete symbol sequence)

– Automatic speech recognition (ASR), Voice Activity Detection (VAD)

Text (discrete symbol sequence) $\rightarrow$ Speech (continuous time series)

– Text-to-speech synthesis (TTS)

Text (discrete symbol sequence) $\rightarrow$ Text (discrete symbol sequence)

– Machine translation (MT)

Speech sequence-to-sequence mapping commonalities

- Variable length sequences
- Highly non-linear relationship
- Increasing quantities of data for training
  - Google Now, Siri, Cortana have gathered 1000s of hours of audio
  - A lot of the data is untranscribed or only has approximate labels
- Increasing diversity in the data
  - broader range of speakers - accents, first language
  - broader range of environmental noises
- Lots of room for improvement still!

Deep Neural Networks are very much part of the solution (cause?)

(Deep) Neural Networks

- Neural networks have increasingly been applied in speech since 2009
  - initially applied to speech recognition [1, 2, 3, 4]
  - "Neural Networks" in title of 8% INTERSPEECH 2015 sessions: feature extraction, modelling, speaker recognition, speech synthesis etc

- But we've been here before haven't we?
  - alternative to GMM-HMMs for ASR in 1980s/early 90s e.g. [5, 6, 7, 8, 9, 10, 11]
  - ✓ smaller footprint than GMM-HMM-based systems
  - × did not perform as well - limited context modelling, adaptation

- What's changed?
  - Significant increase in computing power: CPU and GPU
  - Big data
  - → More powerful networks: more layers (**deep**) and finer targets (**wide**)

Success of neural networks in ASR and TTS

- Speech recognition
  - Systems from Google and IBM reported in [12]

Task	Hours of data	% Word error rate (WER)		
		HMM-DNN	HMM-GMM w/ same data	HMM-GMM w/ more data
Voice Input	5,870	12.3	N/A	16.0
YouTube	1,400	47.6	52.3	N/A
Switchboard	300	12.4	14.5	N/A

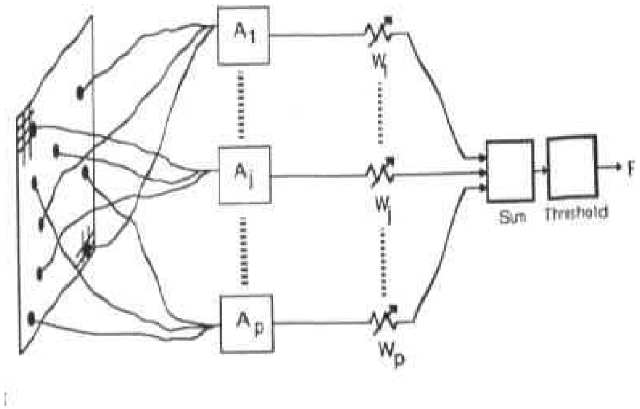
Current best: Switchboard 10.4% using joint CNN/DNN and iVector features [13]

- Parametric speech synthesis [14]
  - Speech samples kindly provided by Heiga Zen, Google

Basics of Neural Networks

Where it started

- Early work by MuCulloch and Pitts [15]
- The Perceptron (Rosenblatt) [16] (early 1960s)

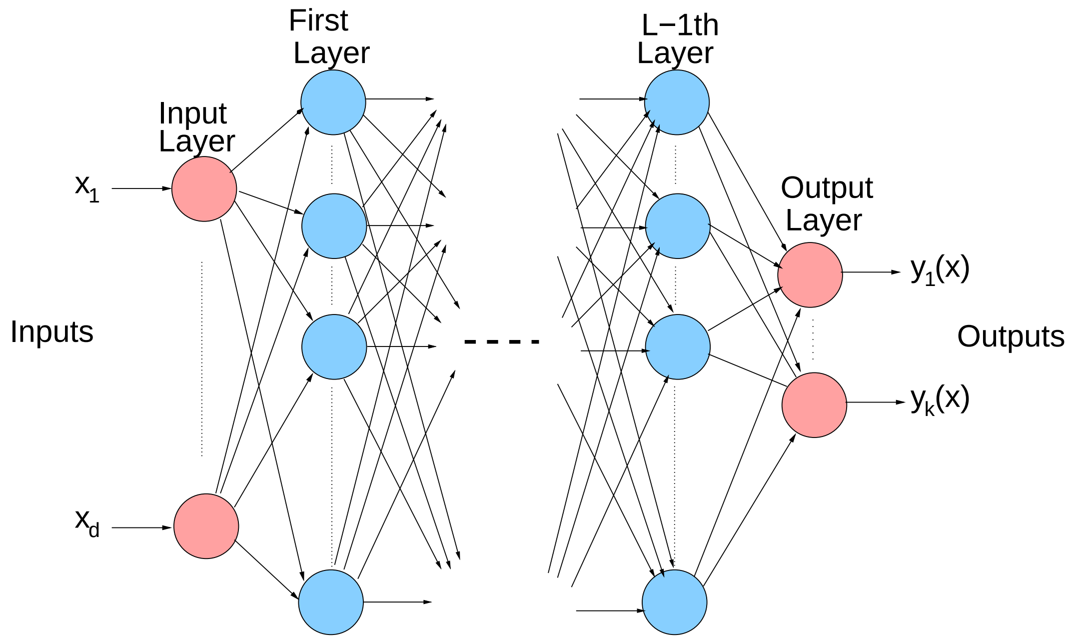


Source: Arvin Calspan Advanced Technology Center; Hecht−Nielsen R. Neurocomputing (Reading, Mass.: Addison−Wesley, 1990)

Source: rutherfordjournal.org

- Mostly halted by publication of “Perceptrons” by Minsky and Papert 1969 [17]
- Error back propagation training for multi-layer perceptrons mid 80s [18]

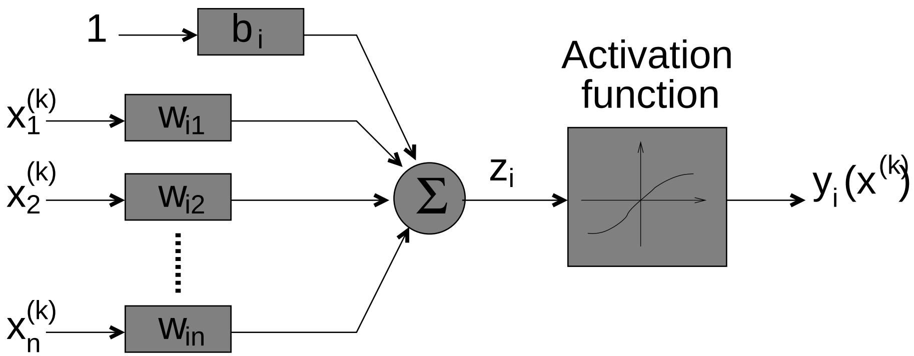
Neural Network



- Aim: map an input vector $\boldsymbol{x}$ into an output vector $\boldsymbol{y}$
  - Non-linear units "neurons" combined into one or more layers
  - Intuition: each layer produces a higher level feature representation and better classifier than its input
  - Combine simple building blocks to design more complex, non-linear systems

Hidden Layer Neuron

- Linearly weighted input is passed to a general activation function
- Assume n units at previous level $(k-1)$: $x_j^{(k)} = y_j(\boldsymbol{x}^{(k-1)})$



$$y_i(\boldsymbol{x}^{(k)}) = \phi(z_i) = \phi(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x}^{(k)} + b_i) = \phi(\sum_{j=1}^{n} w_{ij}x_j^{(k)} + b_i)$$

where $\phi()$ is the activation function

- Note: activation function could be linear BUT then linear net i.e. lose power!

Traditional Activation Functions

- Sigmoid (or logistic regression) function:

$$y_i(\boldsymbol{x}) = \frac{1}{1 + \exp(-z_i)}$$

  Continuous output, $0 \leq y_i(\boldsymbol{x}) \leq 1$

- Softmax (or normalised exponential or generalised logistic) function:

$$y_i(\boldsymbol{x}) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}$$

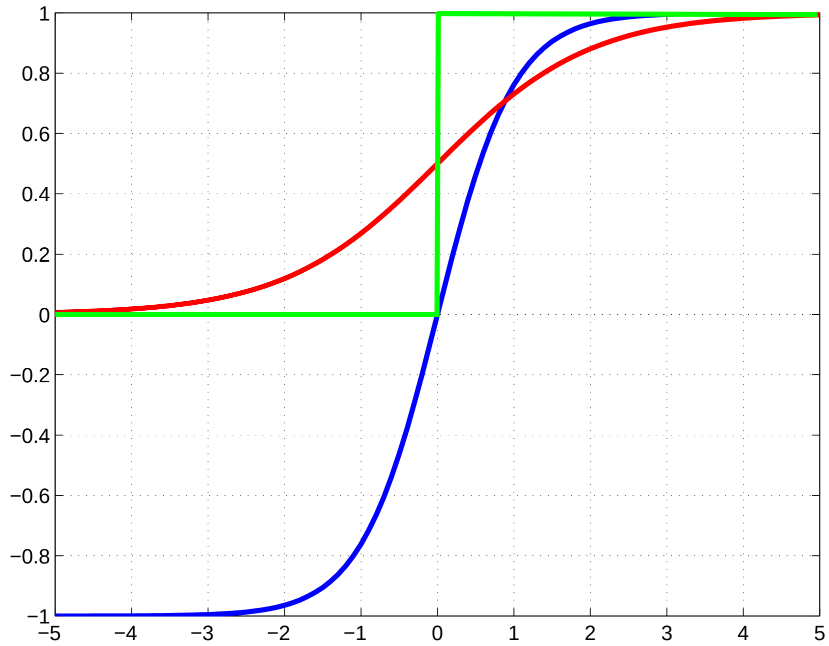
  Positive output, sum of all outputs at current level is 1, $0 \leq y_i(\boldsymbol{x}) \leq 1$

- Hyperbolic tan (tanh) function:

$$y_i(\boldsymbol{x}) = \frac{\exp(z_i) - \exp(-z_i)}{\exp(z_i) + \exp(-z_i)}$$

  Continuous output, $-1 \leq y_i(\boldsymbol{x}) \leq 1$

Activation functions

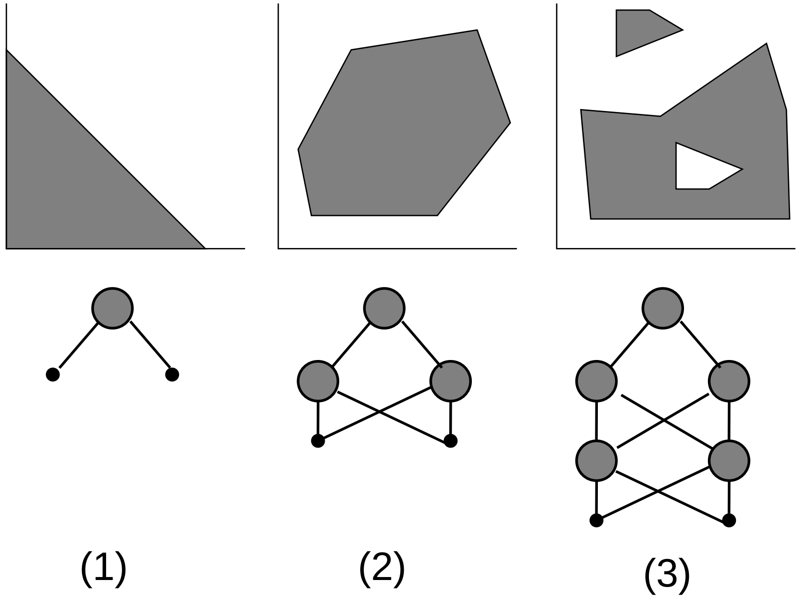


- step activation function (green)
- sigmoid activation function (red)
- tanh activation function (blue)

Sigmoid or softmax often used at output layers as sum-to-one constraint enforced

Possible Decision Boundaries

- Nature of decision boundaries produced varies with network topology
- Using a threshold (step) activation function:



1. Single layer: position a hyperplane in the input space (SLP)
2. Two layers: surround a single convex region of input space
3. Three layers: generate arbitrary decision boundaries

- Sigmoid: arbitrary boundaries with two layers if enough hidden units

Number of Units per Layer

How many units to have in each layer?

- Number of output units = number of output classes
- Number of input units = number of input dimensions
- Number of hidden units - design issue
  - too few - network will not model complex decision boundaries
  - too many - network will have poor generalisation

Training Criteria (1)

Variety of training criteria may be used.

- Assume we have supervised training examples

$$\{\{\boldsymbol{x}_1, \boldsymbol{t}_1\} \ldots, \{\boldsymbol{x}_n, \boldsymbol{t}_n\}\}$$

- Compare outputs $\boldsymbol{y}$ with correct answer $\boldsymbol{t}$ to get error signal
- Least squares error: one of the most common training criteria

$$\begin{aligned} E &= \frac{1}{2}\sum_{p=1}^{n} ||\boldsymbol{y}(\boldsymbol{x}_p) - \boldsymbol{t}_p)||^2 \\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{i=1}^{K} (y_i(\boldsymbol{x}_p) - t_{pi})^2 \end{aligned}$$

Training Criteria (2)

- Cross-Entropy for two classes: consider case when t is binary (softmax output)

$$E = -\sum_{p=1}^{n} \left(t_p \log(y(\boldsymbol{x}_p)) + (1 - t_p) \log(1 - y(\boldsymbol{x}_p))\right)$$

  Goes to zero with the "perfect" mapping

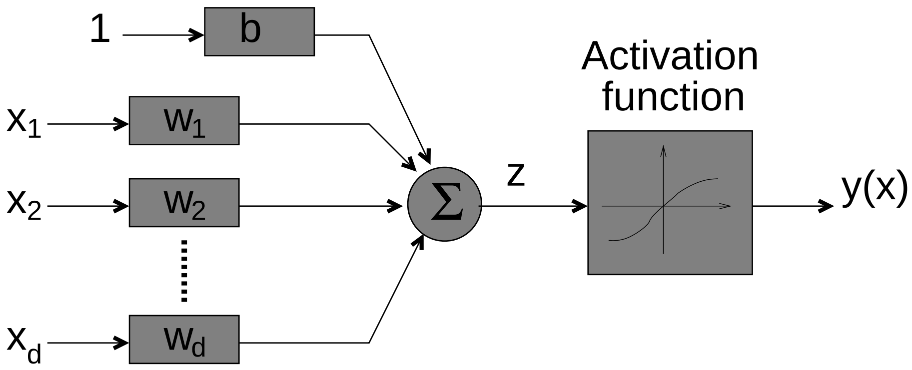
- Cross-Entropy for multiple classes:

$$E = -\sum_{p=1}^{n} \sum_{i=1}^{K} t_{pi} \log(y_i(\boldsymbol{x}_p))$$

  - minimum value is non-zero
  - represents the entropy of the target values

Single Layer Perceptron Training (1)

- Consider single layer perceptron initially



- Minimise (for e.g.) square error between target t_p and current output $y(\boldsymbol{x}_p)$
- Least squares criterion with sigmoid activation function

$$E = \frac{1}{2}\sum_{p=1}^{n}(y(\boldsymbol{x}_p) - t_p)^{\mathsf{T}}(y(\boldsymbol{x}_p) - t_p)) = \sum_{p=1}^{n} E^{(p)}$$

- Simplify notation: single observation $\boldsymbol{x}$, target t, current output $y(\boldsymbol{x})$

Single Layer Perceptron Training (2)

- How does the error change as $y(\boldsymbol{x})$ changes?

$$\frac{\partial E}{\partial y(\boldsymbol{x})} = y(\boldsymbol{x}) - t$$

BUT we want to find the effect of varying the weights

- Calculate effect of changing z on the error using the chain rule

$$\frac{\partial E}{\partial z} = \left(\frac{\partial E}{\partial y(\boldsymbol{x})}\right)\left(\frac{\partial y(\boldsymbol{x})}{\partial z}\right)$$

- What we really want is the change of the error with respect to the weights
  – the parameters that we want to learn

$$\frac{\partial E}{\partial w_i} = \left(\frac{\partial E}{\partial z}\right)\left(\frac{\partial z}{\partial w_i}\right)$$

Single Layer Perceptron Training (3)

- The error function therefore depends on the weight as

$$\frac{\partial E}{\partial w_i} = \left(\frac{\partial E}{\partial y(\boldsymbol{x})}\right)\left(\frac{\partial y(\boldsymbol{x})}{\partial z}\right)\left(\frac{\partial z}{\partial w_i}\right)$$

- Noting that (the bias term b can be treated as the $d+1$ element)

$$\frac{\partial y(\boldsymbol{x})}{\partial z} = y(\boldsymbol{x})(1 - y(\boldsymbol{x}))$$

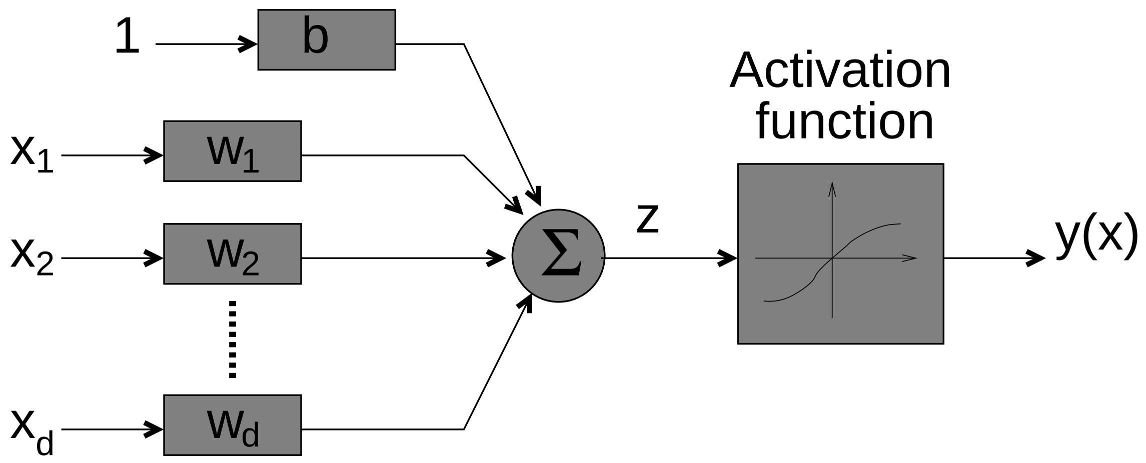
$$\frac{\partial E}{\partial w_i} = (y(\boldsymbol{x}) - t)y(\boldsymbol{x})(1 - y(\boldsymbol{x}))x_i$$

- In terms of the complete training set

$$\boldsymbol{\nabla} E = \sum_{p=1}^{n}(y(\boldsymbol{x}_p) - t_p)y(\boldsymbol{x}_p)(1 - y(\boldsymbol{x}_p))\tilde{\boldsymbol{x}}_p$$

- So for single layer can use gradient descent to find the "best" weight values

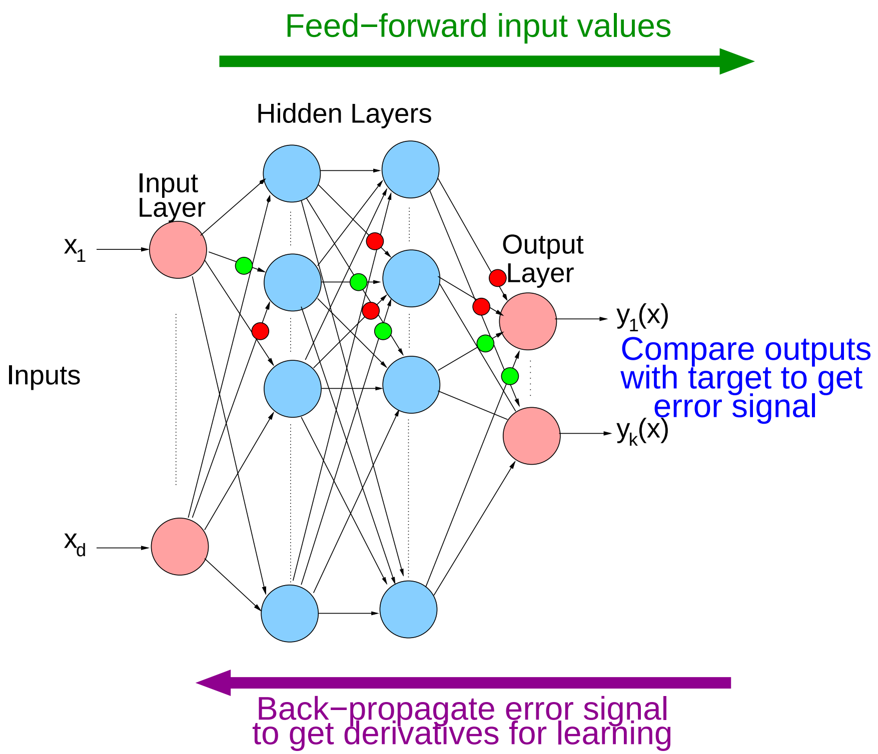
Single Layer Perceptron Training - Review



$$\frac{\partial E}{\partial w_i} = \left(\frac{\partial E}{\partial y(\boldsymbol{x})}\right)\left(\frac{\partial y(\boldsymbol{x})}{\partial z}\right)\left(\frac{\partial z}{\partial w_i}\right)$$

Error Back Propagation Algorithm

- Training Goal: minimise the cost between predicted output and target values
- Error back propagation [18] is an effective way to achieve this



- Use Gradient Descent to optimise the weight values
  - i.e. activation function must be differentiable

Training schemes

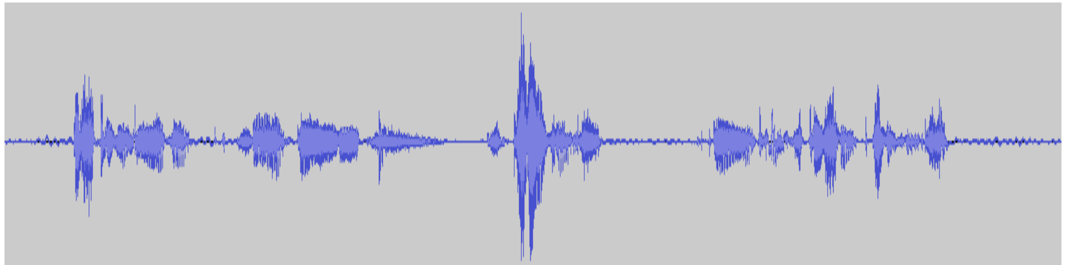
Modes

- Batch - update weights after all training examples seen
- Sequential - update weights after every sample
  Advantages:
  - Don't need to store the whole training database
  - Can be used for online learning
  - In dynamic systems weight updates "track" the system
- Mini-batch - update weights after a subset of examples seen
  Practical compromise:
  - Estimate based on more data than sequential
  - Avoids expensive batch computation if poor current weight values

Voice Activity Detection

Voice Activity Detection

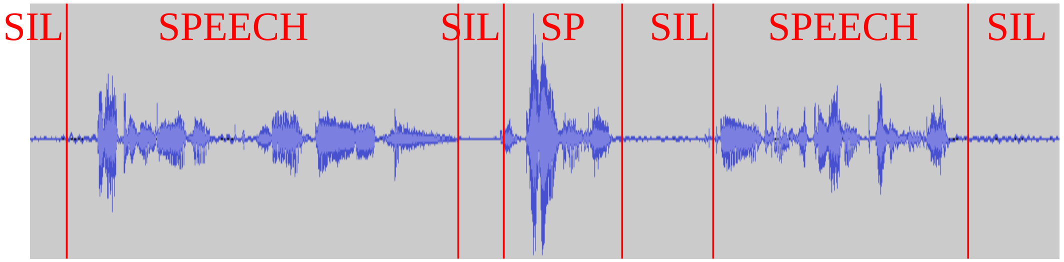
- Detect periods of human speech in an audio signal



Samples from MGB Challenge 2015 [19]

Voice Activity Detection

- Detect periods of human speech in an audio signal



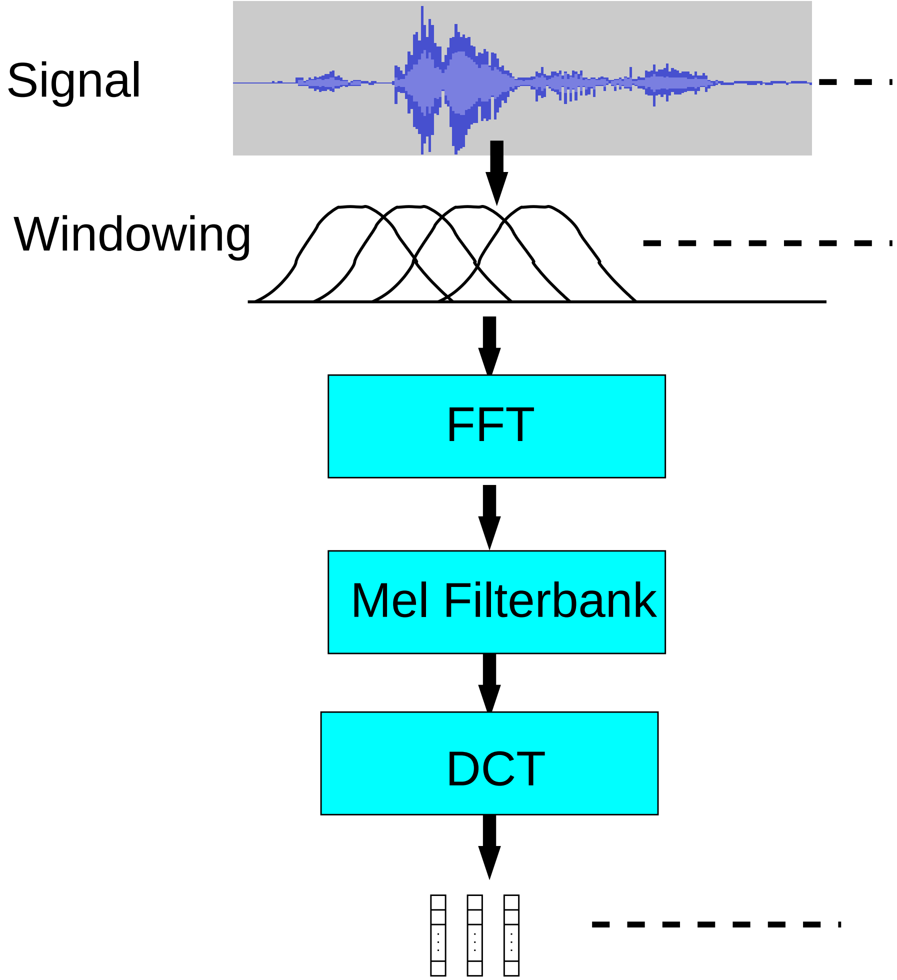
- Sequence classification task
  - 2-class problem: speech or non-speech
- Standard approaches:
  - Unsupervised - threshold against a value e.g. energy, zero-crossing rate
  - Supervised - train a classifier with features such as MFCCs or PLPs e.g. Gaussian mixture models (GMMs), support vector machines

VAD stages

1. Feature extraction
   - compact representation of signal
   - "uncorrelated" to allow diagonal covariance Gaussians

2. Decision making
   - probability of being speech/non-speech computed each frame

3. Hangover
   - smooth decisions
   - 2-state HMM in Viterbi decoding



Gaussian Mixture Models

- Gaussian mixture models (GMMs) are based on (multivariate) Gaussians
  - form of the Gaussian distribution:

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

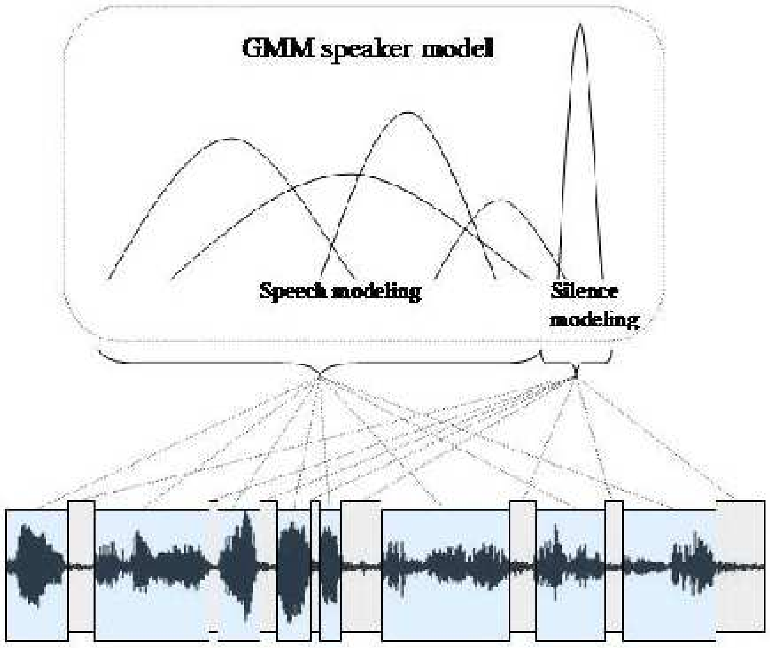
- For GMM each component modelled using a Gaussian distribution

$$p(\boldsymbol{x}) = \sum_{m=1}^{M} P(\mathsf{c}_m)p(\boldsymbol{x}|\mathsf{c}_m) = \sum_{m=1}^{M} P(\mathsf{c}_m)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

  - component prior: $P(\mathsf{c}_m)$
  - component distribution: $p(\boldsymbol{x}|\mathsf{c}_m) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$

- Highly flexible model, able to model wide-range of distributions

GMM-HMM based VAD



Source: Robust Speaker Diarization for Meetings, X.Anguera, Phd Thesis

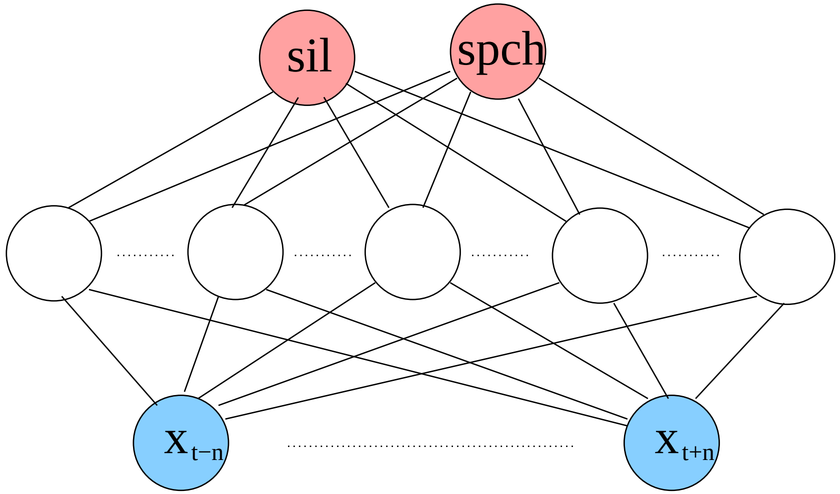
✓ Work well under stationary noise conditions

× Do not generalise to diverse domains e.g. meetings, YouTube

DNN based VAD

- Replace GMM probability density function in HMM with DNN output [20]
  - First must convert output posteriors to likelihoods

$$p(\boldsymbol{x}_t|\texttt{spch}) = \frac{P(\texttt{spch}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{P(\texttt{spch})}$$



✓ Significantly more accurate in challenging environments
e.g. 20% frame-wise error rate on YouTube vs 40% GMM system [21]

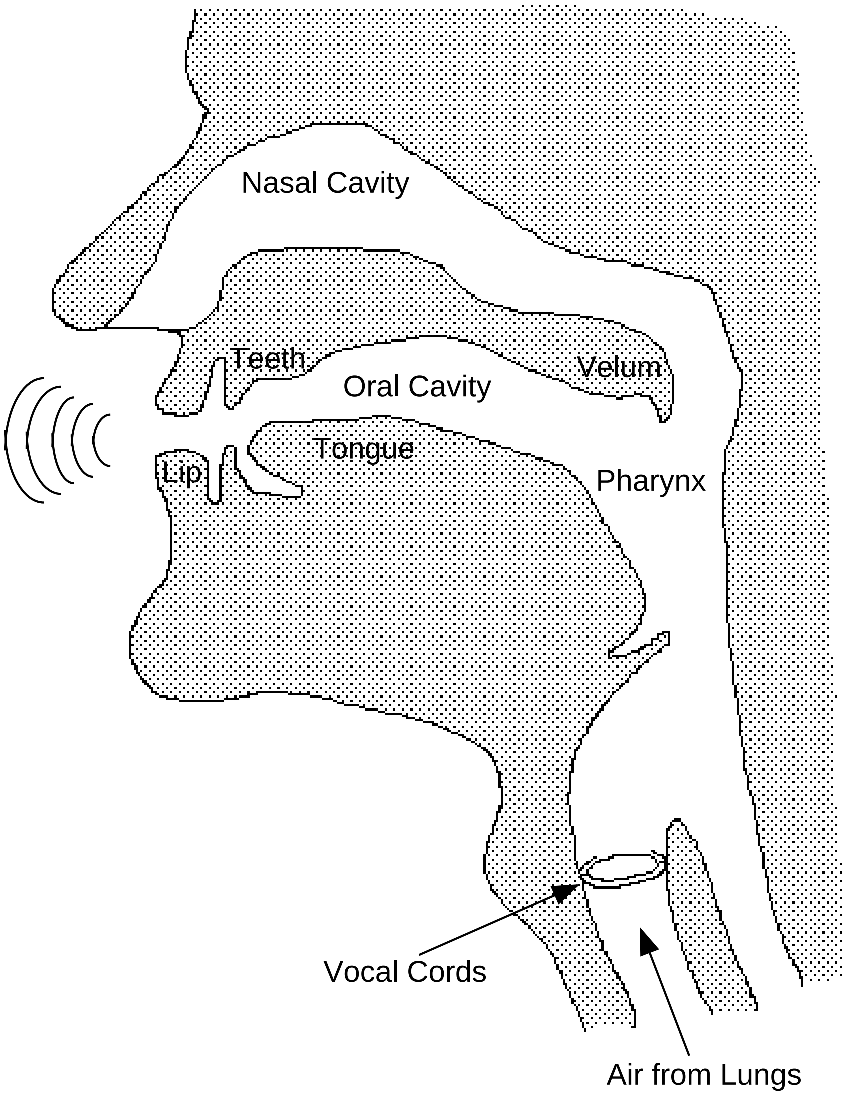
DNN-based VAD - training considerations

- Input features
  - Can use same MFCC or PLP features as for GMM
  - Gains shown when extending context [21]
  - Filterbanks show further gains [22]
- Targets
  - Each training frame is tagged as speech/non-speech
  - Following DNN training, data can be realigned including unlabelled data
- Example system: Cambridge University MGB Challenge 2015 VAD [22]
  - Input: 40-d filterbanks, 55 frames (± 27)
  - Layers: 1000 $\times$ 200^5 $\times$ 2
  - Activation functions: sigmoid
  - Targets: alignments derived from lightly supervised recognition
  - Training criterion: frame-based cross-entropy (CE)

Language Identification

Automatic Speech Recognition

Speech Production



- Excitation source
  - vocal cords vibrate producing quasi-periodic sounds (voiced sounds)
  - turbulence caused by forcing air through a constriction in the vocal tract (fricative sounds)
- Acoustic tube
  - articulators move:
    alter the shape of the vocal tract
    enable/disable nasal cavity
  - co-articulation effect.
- Speech
  - sound pressure wave.

Automatic Speech Recognition - Theory

- Speech recognition based on Bayes' Decision Rule

$$\hat{\boldsymbol{w}} = \max_{\boldsymbol{w}} \{P(\boldsymbol{w}|\mathbf{O})\}$$

$\mathbf{O} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ and $\boldsymbol{w} = \{w_1, \ldots, w_L\}$

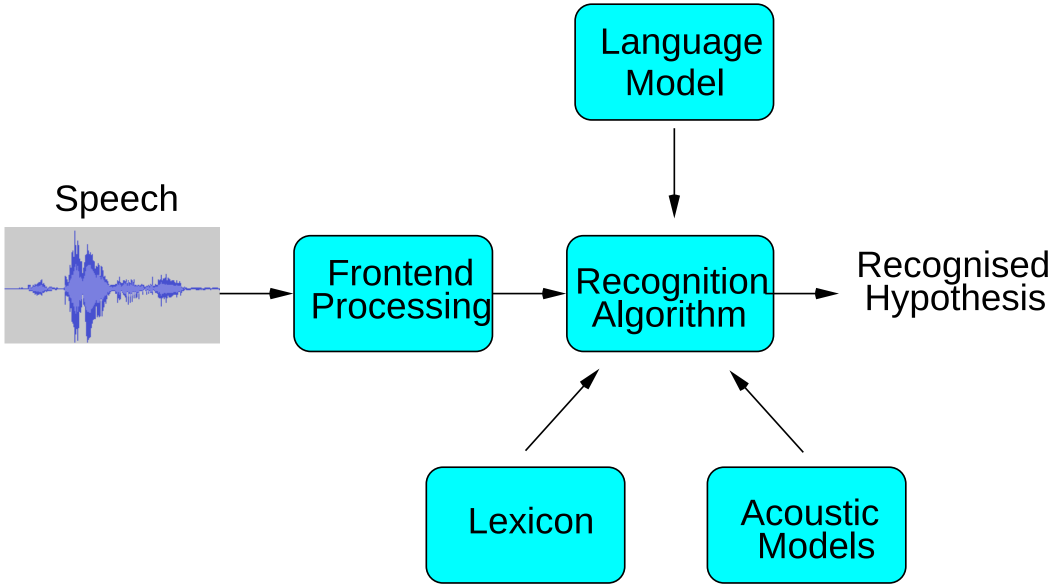
- Two forms of classifier used:
  - Generative model: model joint distribution $p(\mathbf{O}, \boldsymbol{w})$

$$P(\boldsymbol{w}|\mathbf{O}) = \frac{p(\mathbf{O}, \boldsymbol{w})}{p(\mathbf{O})} \propto p(\mathbf{O}|\boldsymbol{w})P(\boldsymbol{w})$$

  - Discriminative model: directly model posterior distribution $P(\boldsymbol{w}|\mathbf{O})$

Machine Learning underpins all ASR systems

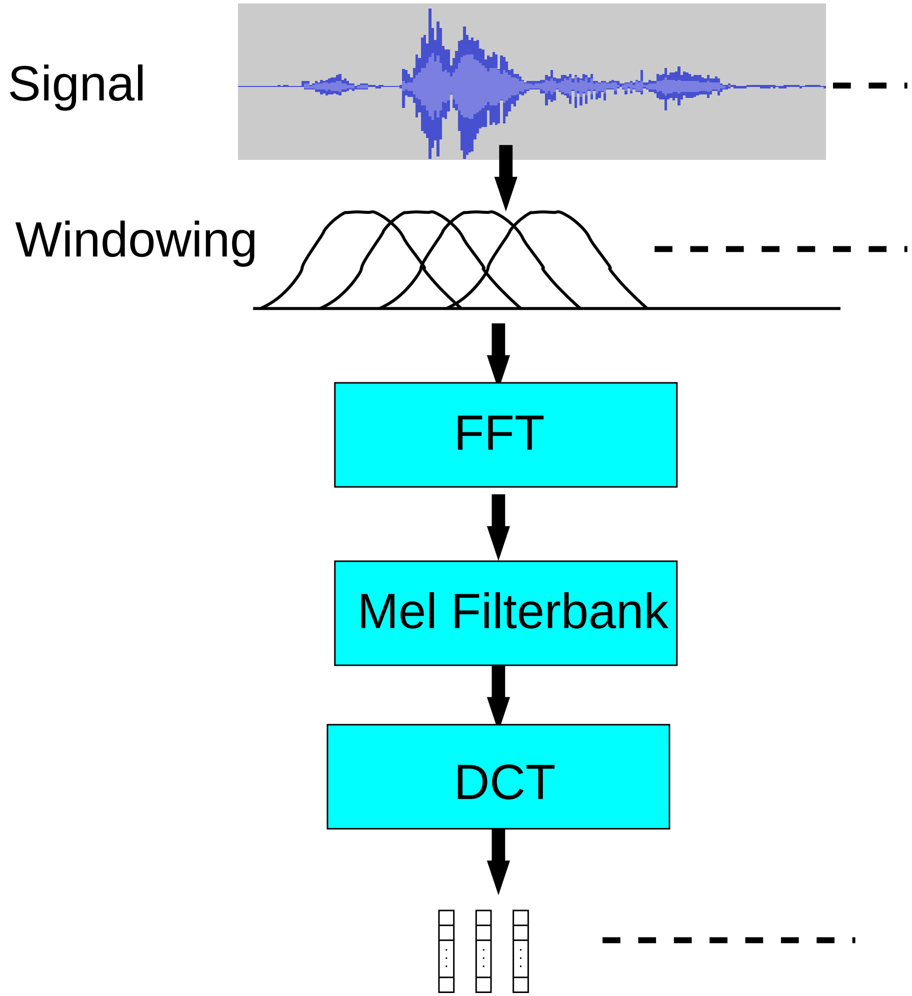
Automatic Speech Recognition - Modules



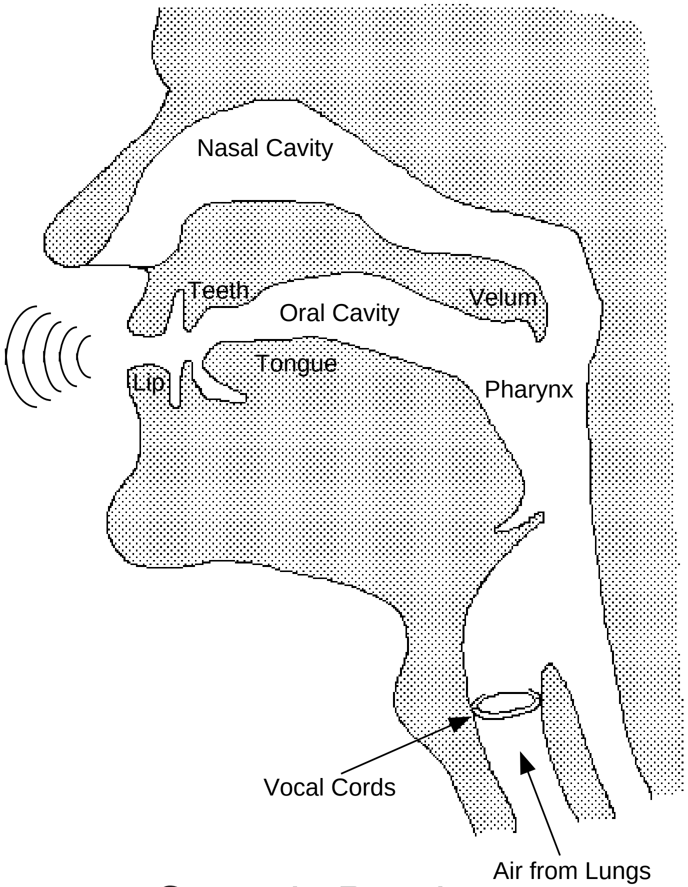
- Front-end processing: transforms waveform into *acoustic vectors*
- Acoustic model: probability of observations given a word sequence
- Lexicon: maps from word to phone sequence
- Language model: computes the prior probability of any word sequence

Statistical approaches used to combine information sources

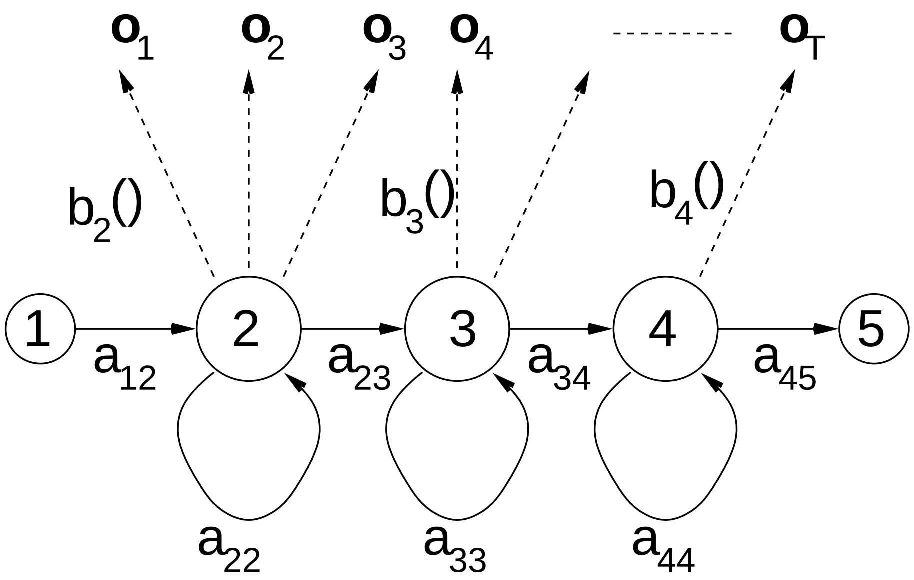
Front End Processing



Acoustic Modelling



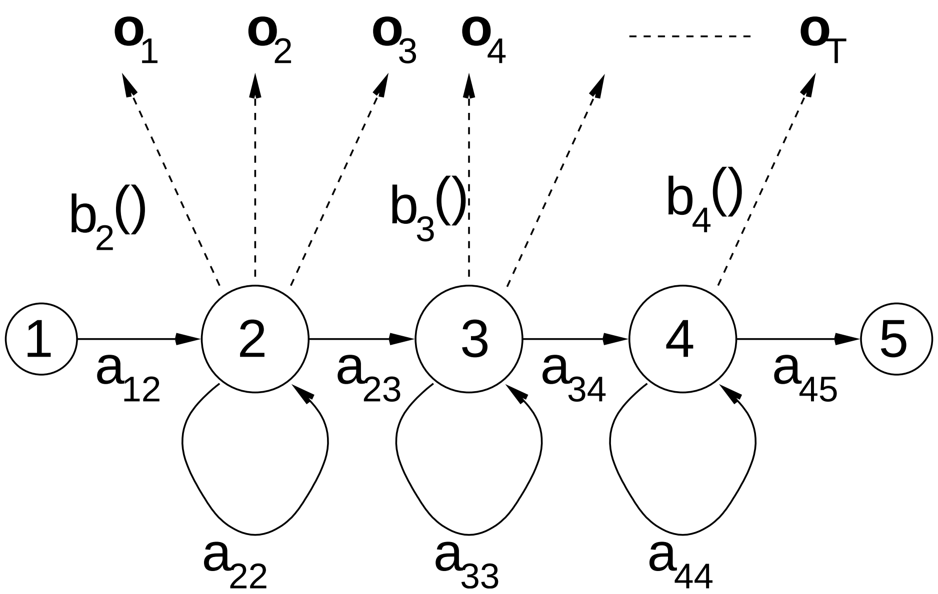
(a) Speech Production



(b) HMM Generative Model

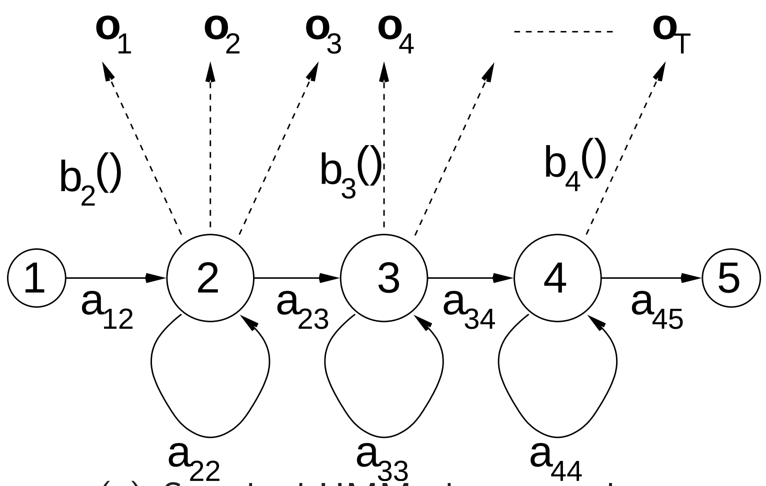
- Not modelling the human production process!

Hidden Markov Model "Production"

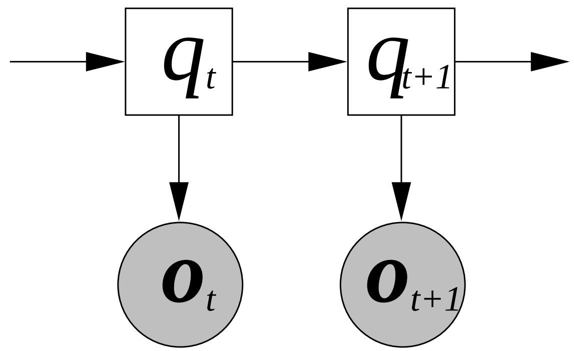


- State evolution process
  - discrete state transition after each "observation"
  - probability of entering a state only dependent on the previous state
- Observation process
  - associated with each state is a probability distribution
  - observations are assumed independent given the current state
- Speech representation
  - feature vector every 10ms

Hidden Markov Model



(c) Standard HMM phone topology



(d) HMM Dynamic Bayesian Network

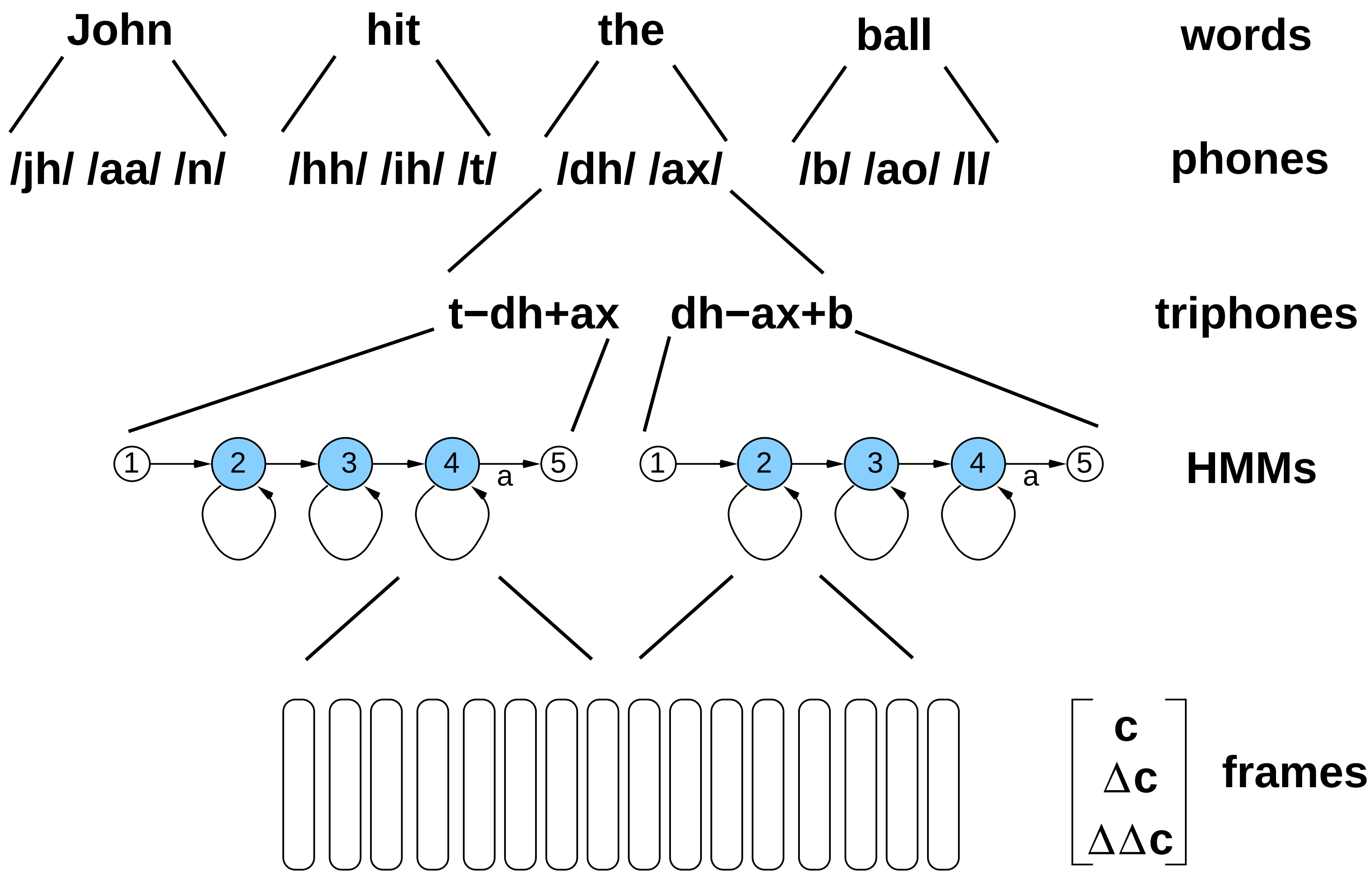
- The likelihood of the data is

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} P(\boldsymbol{q}) p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T | \boldsymbol{q}) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} P(q_0) \prod_{t=1}^{T} P(q_t | q_{t-1}) p(\boldsymbol{x}_t | q_t)$$

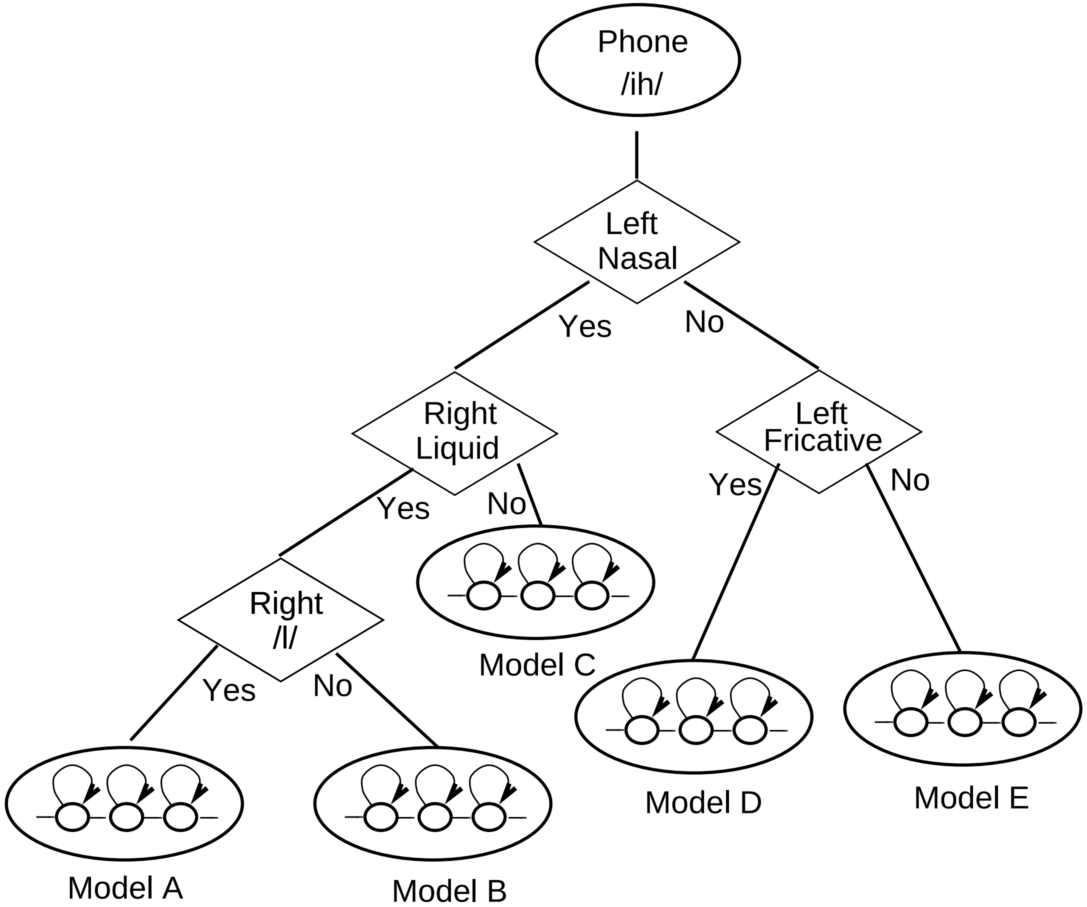
$\boldsymbol{q} = \{q_0, \ldots, q_{T+1}\}$ and $\boldsymbol{Q}_T$ is all possible state sequences for T observations

- Poor model of the speech process - piecewise constant state-space.

HMM Acoustic Units



State Tying - Decision Tree



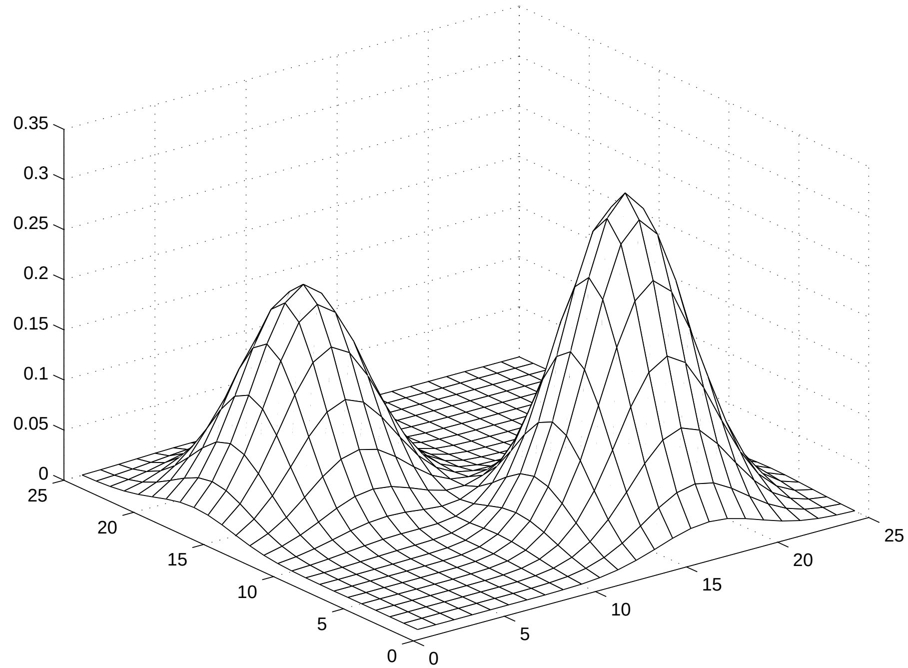
State Output Distribution: Gaussian Mixture Model

A common form of distribution associated with each state:

- the Gaussian mixture model (or mixture of Gaussians).

- linear combination of components

$$p(\boldsymbol{x}_t) = \sum_{m=1}^{M} c^{(m)} \mathcal{N}(\boldsymbol{x}_t, \boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)})$$



- Good modelling power:
  - implicitly models variability
- No constraints on component choice

HMM Training using EM

- Need to train HMM model parameters, $\boldsymbol{\lambda}$, on 100s of millions of frames
  - transition probabilities
  - state output distribution

- Standard training criterion for generative models: Maximum Likelihood

$$\mathcal{F}_{\tt ml}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\log(p(\mathbf{O}^{(r)}|\mathbf{w}_{\tt ref}^{(r)};\boldsymbol{\lambda}))$$

  - yields most likely model parameters to generate training data!

- Challenging to handle vast amounts of data
  - Expectation Maximisation (EM) offers a solution

HMM Training using EM

- EM an iterative scheme involving two stages:
  - Expectation: accumulate statistics given current model parameters
  - Maximisation: estimate new model parameters

- Update formulae for GMM state output distributions

$$\boldsymbol{\mu}_j^{[l+1]} = \frac{\sum_{t=1}^{T} \gamma_j^{[l]}(t)\boldsymbol{x}_t}{\sum_{t=1}^{T} \gamma_j^{[l]}(t)}$$

$$\boldsymbol{\Sigma}_j^{[l+1]} = \frac{\sum_{t=1}^{T} \gamma_j^{[l]}(t)\boldsymbol{x}_t\boldsymbol{x}_t^{\mathsf{T}}}{\sum_{t=1}^{T} \gamma_j^{[l]}(t)} - \boldsymbol{\mu}_j^{[l+1]}\boldsymbol{\mu}_j^{[l+1]\mathsf{T}}$$

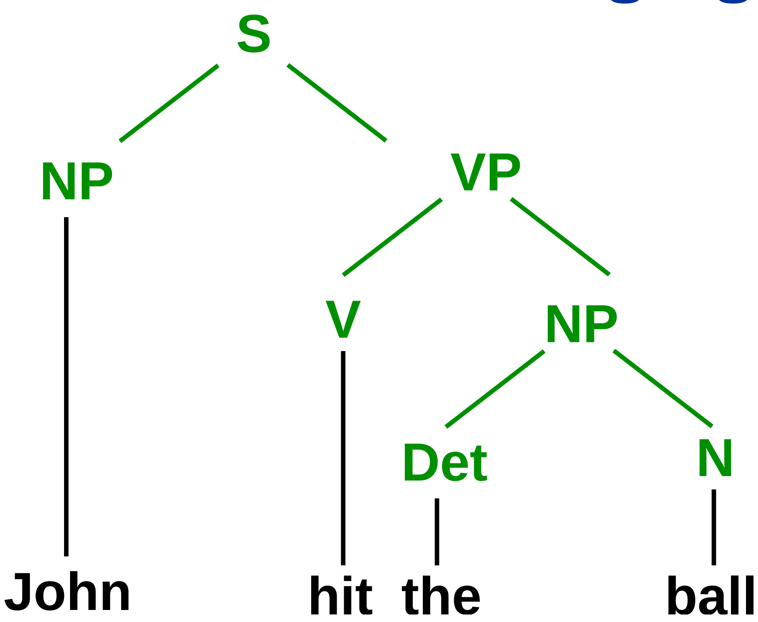
where

$$\gamma_j^{[l]}(t) = P(q_t = \mathrm{s}_j | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \boldsymbol{\lambda}^{[l]})$$

Advantages of EM training

- EM is one of the reasons GMM-HMM systems dominated for many years
  - guaranteed not to decrease log-likelihood at each iteration
  - expectation stage can be parallelised
- Parallelising the expectation stage crucial
  - Enables handling of vast quantities of data
  - Can distribute across many cheap machines
- Would like ASR system to run in real-time
  - HMM structure enables this - Viterbi algorithm

Language Model



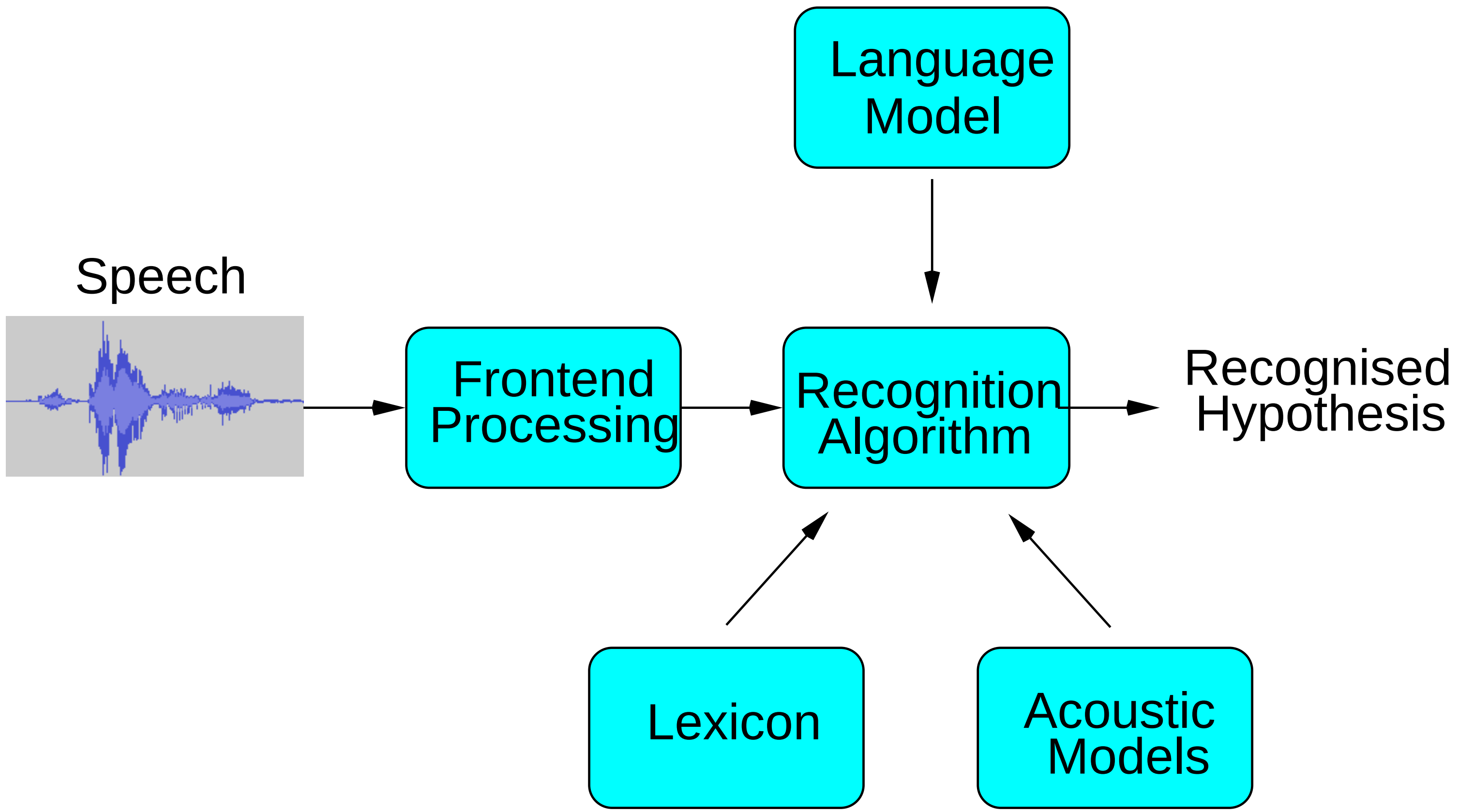
(e) Syntactic Parse Tree

P(John hit the ball) =
P(John) x
P(hit | John) x
P(the | John hit) x
P(ball| hit the)

(f) Trigram Model

- Syntactic/semantic information important
  - but hard to model robustly (especially for conversational style speech)
- Simple n-gram model-used: $P(w_1 w_2 ... w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-2} w_{i-1})$
  - don't care about structure - just the probability - discuss later

Automatic Speech Recognition - Modules



Recognition Algorithm - Viterbi

- An important technique for HMMs (and other models) is the Viterbi Algorithm
  - here the likelihood is approximated as (ignoring dependence on class ω)

$$p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T, \boldsymbol{q}) \approx p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T, \hat{\boldsymbol{q}})$$

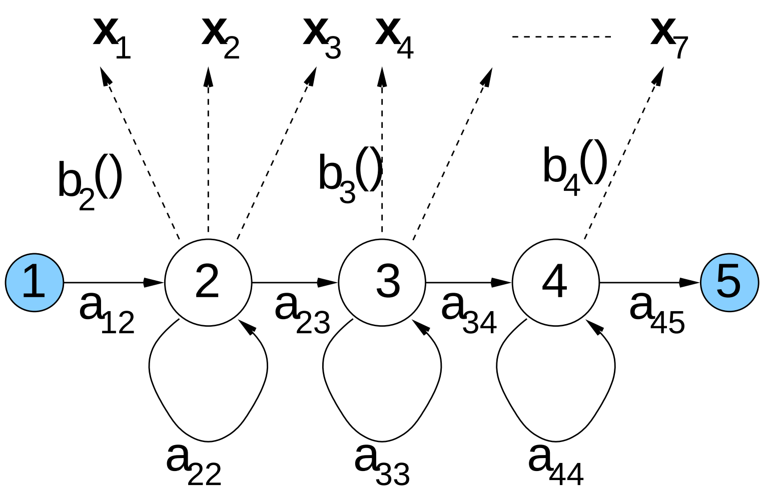
  where

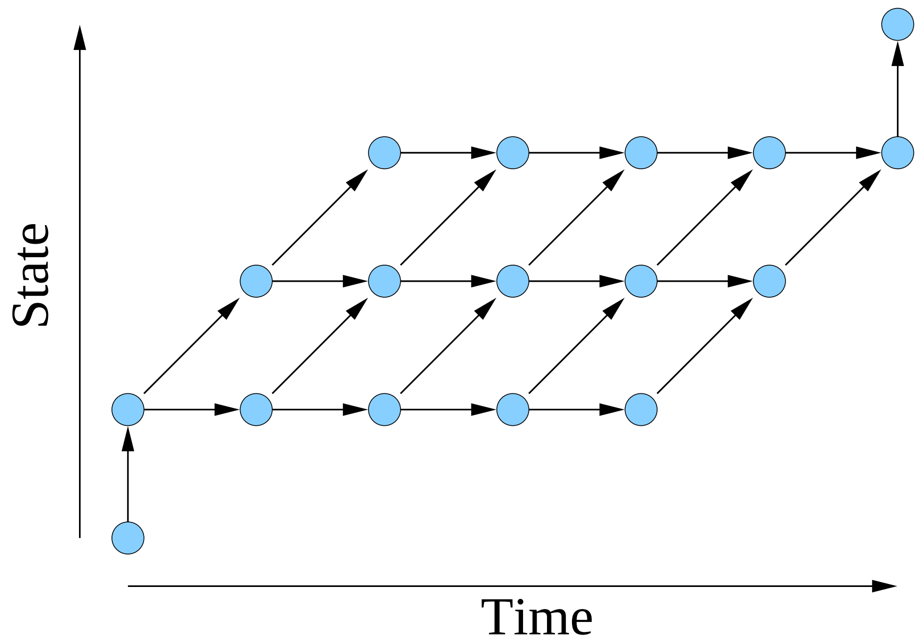
$$\hat{\boldsymbol{q}} = \{\hat{q}_0, \dots, \hat{q}_{T+1}\} = \underset{\boldsymbol{q} \in \boldsymbol{Q}_T}{\operatorname{argmax}} \{p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T, \boldsymbol{q})\}$$

- This yields:
  - an approximate likelihood (lower bound) for the model
  - the best state-sequence through the discrete-state space

Viterbi Algorithm

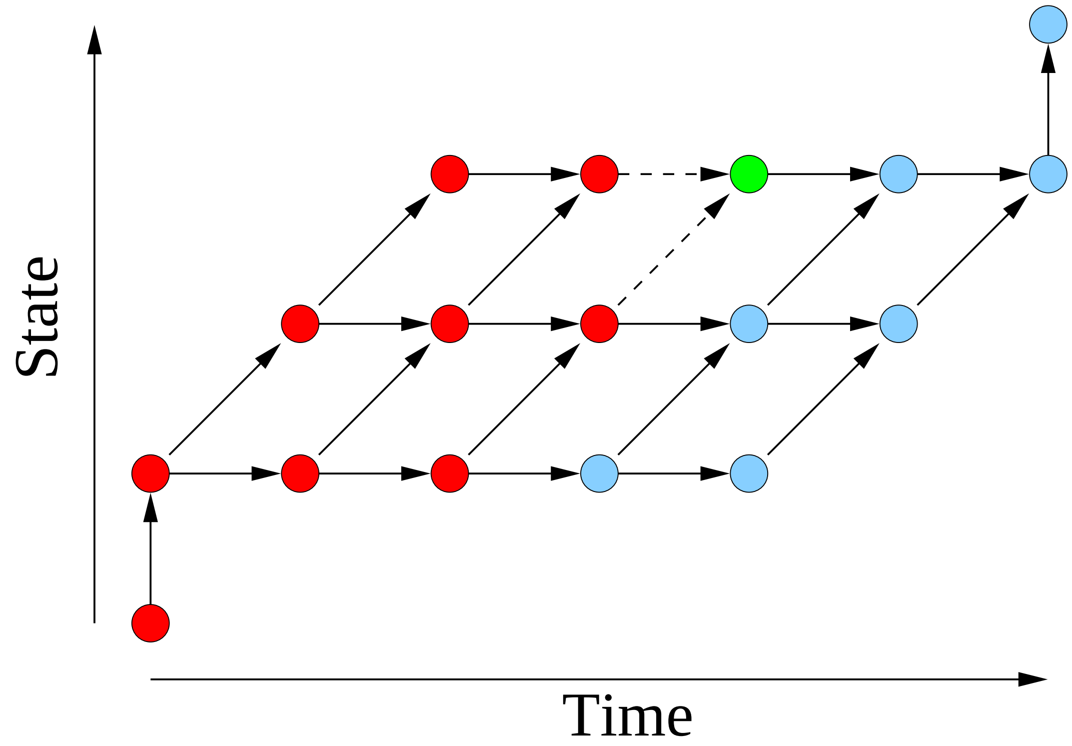
- Need an efficient approach to obtaining the best state-sequence, $\hat{\boldsymbol{q}}$,
  - simply searching through all possible state-sequences impractical ...





- Consider generating the observation sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_7$
  - HMM topology - 3 emitting states with strict left-to-right topology (left)
  - representation of all possible state sequences on the right

Best Partial Path to a State/Time



- Red possible partial paths
- Green state of interest

- Require best partial path to state $\mathtt{s}_4$ at time 5 (with associated cost $\phi_4(5)$)
  - cost of moving from state $\mathtt{s}_3$ and generating observation $\boldsymbol{x}_5$: $\log(a_{34}b_4(\boldsymbol{x}_5))$
  - cost of staying in state $\mathtt{s}_4$ and generating observation $\boldsymbol{x}_5$: $\log(a_{44}b_4(\boldsymbol{x}_5))$
- Select "best: $\phi_4(5) = \max\left\{\phi_3(4) + \log(a_{34}b_4(\boldsymbol{x}_5)), \phi_4(4) + \log(a_{44}b_4(\boldsymbol{x}_5))\right\}$

Viterbi Algorithm for HMMs

- The Viterbi algorithm for HMMs can then be expressed as:
  - Initialisation: $(\texttt{LZERO}= \log(0))$
    $\phi_1(0) = 0.0, \quad \phi_j(0) = \texttt{LZERO}, 1 < j < N,$
    $\phi_1(t) = \texttt{LZERO}, 1 \leq t \leq T$
  - Recursion:
    $\texttt{for } t = 1, \ldots, T$
    $\quad \texttt{for } j = 2, \ldots, N-1$
    $$\phi_j(t) = \max_{1 \leq k < N} \{\phi_k(t-1) + \log(a_{kj})\} + \log(b_j(\boldsymbol{x}_t))$$
  - Termination:
    $$\log(p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \hat{\boldsymbol{q}})) = \max_{1<k<N} \{\phi_k(T) + \log(a_{kN})\}$$
- Can also store the best previous state to allow best sequence $\hat{\boldsymbol{q}}$ to be found.

Discriminative Training Criteria

- Bayes' decision rule yields the minimum probability of error if:
  - infinte training data
  - models have the correct form
  - appropriate training criterion

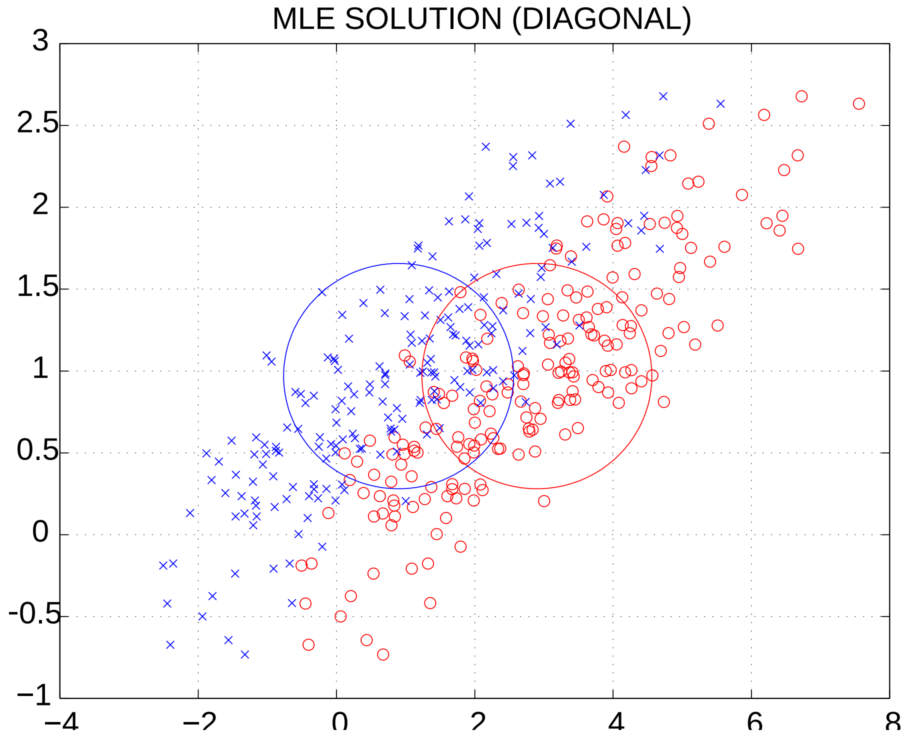
None of these are true for ASR!

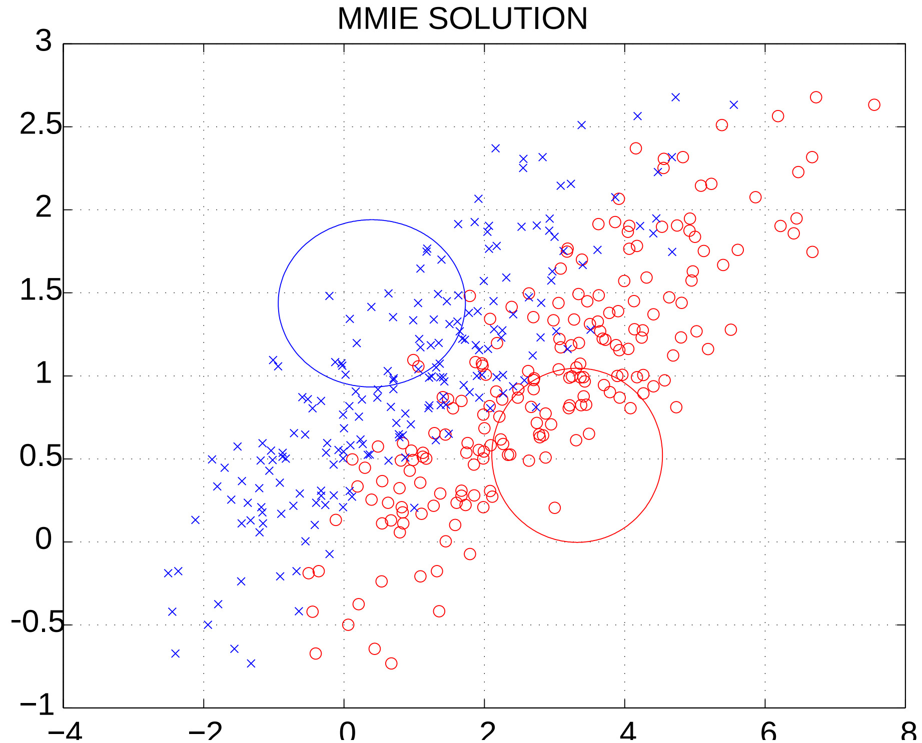
- Motivates other discriminative criteria
  - use discrimative criteria to train generative models
  - ML people not that happy with use and term!

- Forunately schemes relared to EM can still be used
  - large scale discriminative training common for ASR
  - acoustic model still an HMM - Viterbi still possible

Simple MMIE Example

- HMMs are not the correct model - discriminative criteria a possibility





- Discrimnative criteria a function of posteriors $P(\mathbf{w}|\mathbf{O};\boldsymbol{\lambda})$
  – NOTE: same generative model, and conditional independence assumptions

Discriminative Training Criteria

- Discriminative training criteria commonly used to train HMMs for ASR
  - Maximum Mutual Information (MMI) [23, 24]: maximise

$$\mathcal{F}_{\tt mmi}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\log(P(\mathbf{w}_{\tt ref}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda}))$$

  - Minimum Classification Error (MCE) [25]: minimise

$$\mathcal{F}_{\tt mce}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\left(1+\left[\frac{P(\mathbf{w}_{\tt ref}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})}{\sum_{\mathbf{w}\neq\mathbf{w}_{\tt ref}^{(r)}}P(\mathbf{w}|\mathbf{O}^{(r)};\boldsymbol{\lambda})}\right]^{\varrho}\right)^{-1}$$

  - Minimum Bayes' Risk (MBR) [26, 27]: minimise

$$\mathcal{F}_{\tt mbr}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{\mathbf{w}}P(\mathbf{w}|\mathbf{O}^{(r)};\boldsymbol{\lambda})\mathcal{L}(\mathbf{w},\mathbf{w}_{\tt ref}^{(r)})$$

MBR Loss Functions for ASR

- Sentence (1/0 loss):

$$\mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) = \begin{cases} 1; & \mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}^{(r)} \\ 0; & \mathbf{w} = \mathbf{w}_{\mathtt{ref}}^{(r)} \end{cases}$$

When $\varrho = 1$, $\mathcal{F}_{\mathtt{mce}}(\boldsymbol{\lambda}) = \mathcal{F}_{\mathtt{mbr}}(\boldsymbol{\lambda})$

- Word: directly related to minimising the expected Word Error Rate (WER)
  - normally computed by minimising the Levenshtein edit distance.
- Phone/State: consider phone/state rather word loss
  - improved generalisation as more “errors” observed
  - this is known as Minimum Phone Error (MPE) training [28, 29].
- Hamming (MPFE): number of erroneous frames measured at the phone level

Summary of Standard ASR Systems

- HMMs
  - efficiency of model training/decoding
  - approximate approach to modelling the signal
  - has limitations on features that can be used due to GMMs

- GMMs
  - OK but make lots of assumptions about feature vector
    - decorrelated and Gaussian

References

[1] A.R. Mohamed, G.E. Dahl, and G. Hinton, "Deep belief networks for speech recognition," in *Proc. NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[2] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real world speech recognition," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[3] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with continuous speech recognition with context-dependent DBN-HMMs," in *Proc. ICASSP*, 2011.

[4] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[5] A. Waibel et al., "Phoneme recognition using time-delay neural networks," *IEEE Trans. Speech and Audio Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[6] K.J. Lang, A. Waibel, and G.E. Hinton, "A time-delay neural network architecture for isolated word recognition," *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[7] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden Markov models," in *Proc. ICASSP*, 1990.

[8] N. Morgan and H.A. Bourlard, "Neural networks for statistical recognition of continuous speech," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 742–772, 1995.

[9] S. Renals, N. Morgan, H. Bourlard, and H. Franco, "Connectionist probabilitity estimation in the DECIPHER speech recognition system," in *Proc. ICASSP*, 1992.

[10] S. Renals, N. Morgan, H. Bourlard, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[11] A.J. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, 1994.

[12] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[13] H. Soltau, G. Saon, and T.N. Sainath, "Joint Training of Convolutional and Non-Convolutional Neural Networks," in *Proc. ICASSP*, 2014.

[14] Zhen-Hua Ling, Shiyin Kang, Heiga Zen, Andrew Senior, Mike Schuster, Xiao-Jun Qian, Helen Meng, and Li Deng, "Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends," *IEEE Signal Processing Magazine*, vol. 32, pp. 35–52, 2015.

[15] W.S. McCulloch and W.Pitts, ," *The bulletin of mathematical biophysics*, 1943.

[16] F.Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, 1962.

[17] Marvin Minsky and Seymour A. Papert, *Perceptrons*, MIT Press, 1969.

[18] David E. Rumelhart, Geoffery E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[19] Peter Bell, Mark Gales, Thomas Hain, Jonathan Kilgour, Pierre Lanchantin, Xunying Liu, Andrew McParland, Steve Renals, Oscar Saz, Mirjam Wester, and Philip Woodland, "The MGB Challenge: Evaluating Multi-Genre Broadcast Media Recognition," in *To appear Proc. ASRU*, 2015.

[20] John Dines, Jithendra Vepa, and Thomas Hain, "The segmentation of multi-channel meeting recordings for automatic speech recognition," in *Proc. ICSLP*, 2006.

[21] Neville Ryant, Mark Liberman, and Jiahong Yuan, "Speech Activity Detection on YouTube Using Deep Neural Networks," in *Proc. INTERSPEECH*, 2013.

[22] Philip Woodland, Xunying Liu, Yanmin Qian, Chao Zhang, Mark Gales, Penny Karanasou, Pierre Lanchantin, and Linlin Wang, "Cambridge University Transcription Systems for the Multi-Genre Broadcast Challenge," in *To appear Proc. ASRU*, 2015.

[23] P.S. Gopalakrishnan, D. Kanevsky, A. Nádas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Trans. Information Theory*, 1991.

[24] P. C. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech & Language*, vol. 16, pp. 25–47, 2002.

[25] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Transactions on Signal Processing*, 1992.

[26] J. Kaiser, B. Horvat, and Z. Kacic, "A novel loss function for the overall risk criterion based discriminative training of HMM models," in *Proc. ICSLP*, 2000.

[27] W. Byrne, "Minimum Bayes risk estimation and decoding in large vocabulary continuous speech recognition," *IEICE Special Issue on Statistical Modelling for Speech Recognition*, 2006.

[28] D. Povey and P. C. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Orlando, FL, May 2002.

[29] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambridge University, 2004.