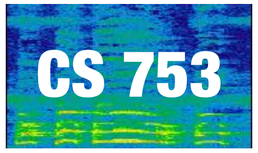
Acoustic Modeling:
Tied-state HMMs & DNN-based models

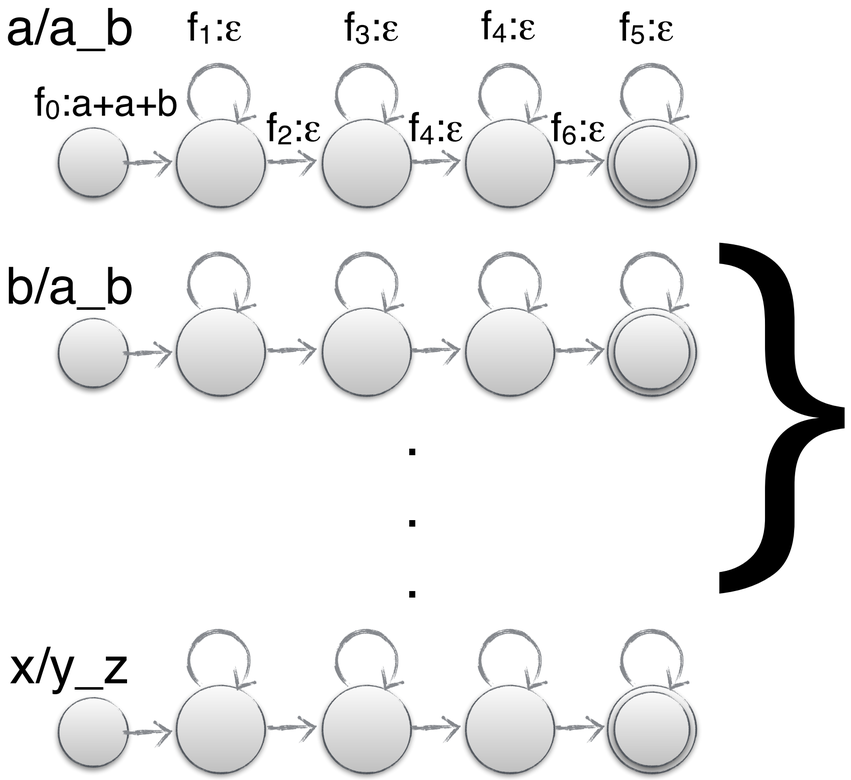
Lecture 7

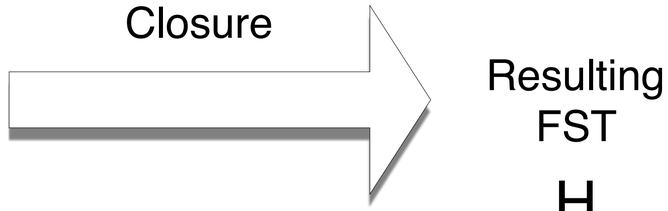


Instructor: Preethi Jyothi

Recall: Acoustic Model

H





Resulting FST

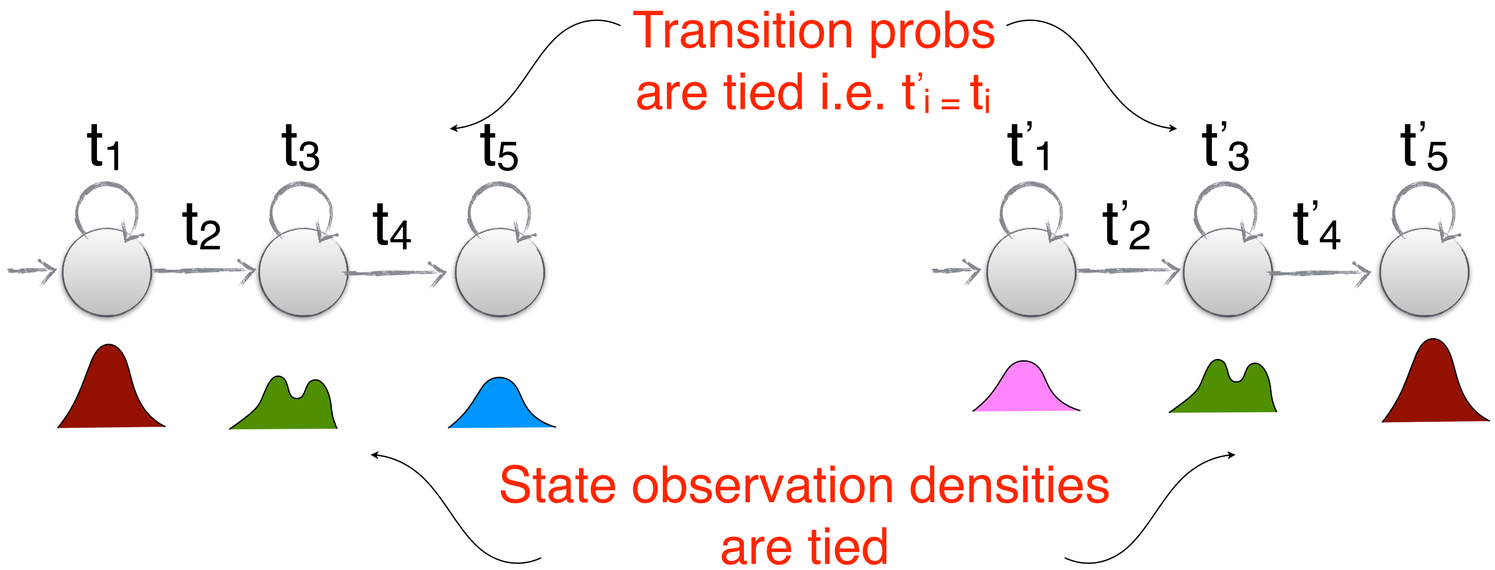
H

Triphone HMM Models

- Each phone is modelled in the context of its left and right neighbour phones
  - Pronunciation of a phone is influenced by the preceding and succeeding phones. E.g. The phone [p] in the word “peek” : p iy k” vs. [p] in the word “pool” : p uw l
- Number of triphones that appear in data ≈ 1000s or 10,000s
- If each triphone HMM has 3 states and each state generates m-component GMMs ($m \approx 64$), for d-dimensional acoustic feature vectors ($d \approx 40$) with Σ having d^2 parameters
  - Hundreds of millions of parameters!
- Insufficient data to learn all triphone models reliably. What do we do? Share parameters across triphone models!

Parameter Sharing

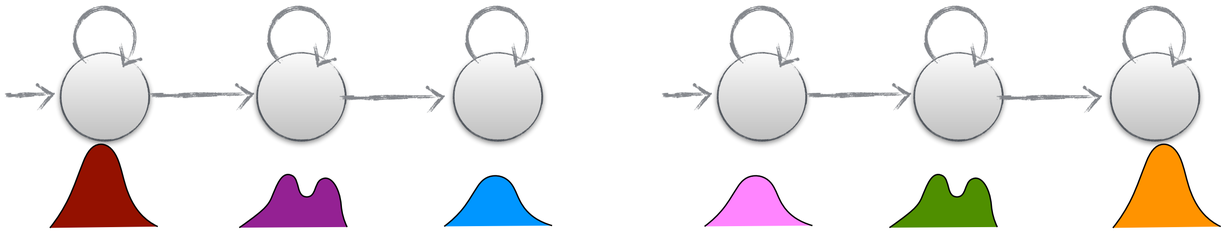
- Sharing of parameters (also referred to as “parameter tying”) can be done at any level:
  - Parameters in HMMs corresponding to two triphones are said to be tied if they are identical



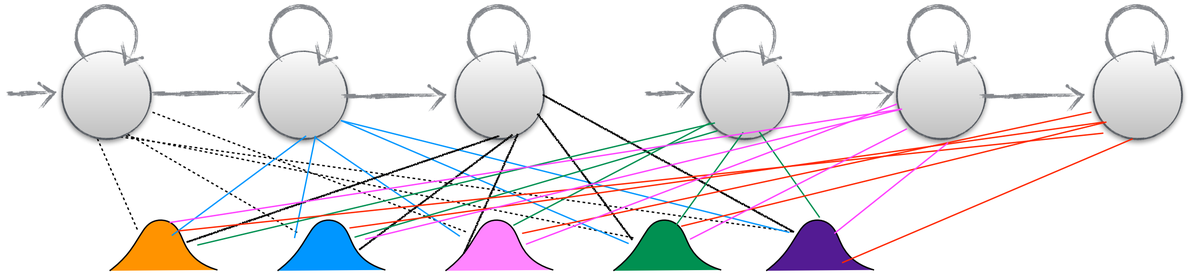
- More parameter tying: Tying variances of all Gaussians within a state, tying variances of all Gaussians in all states, tying individual Gaussians, etc.

1. Tied Mixture Models

- All states share the same Gaussians (i.e. same means and covariances)
- Mixture weights are specific to each state



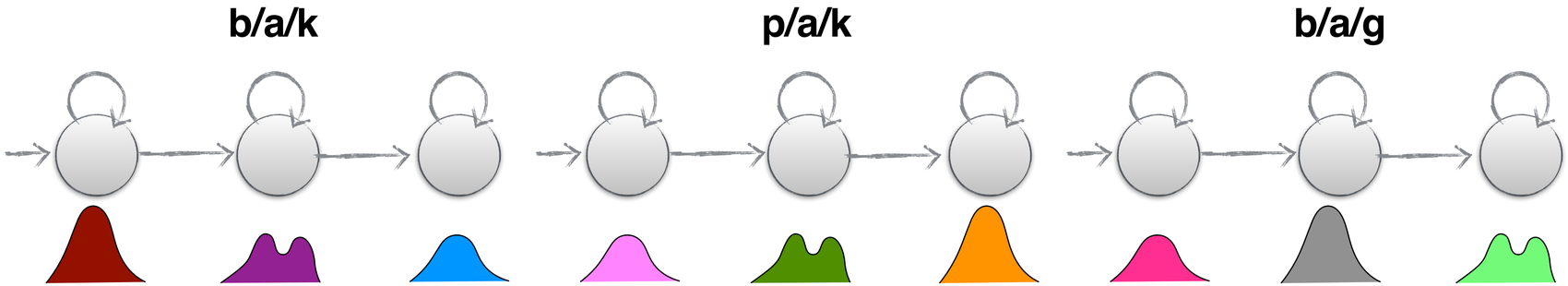
Triphone HMMs (No sharing)



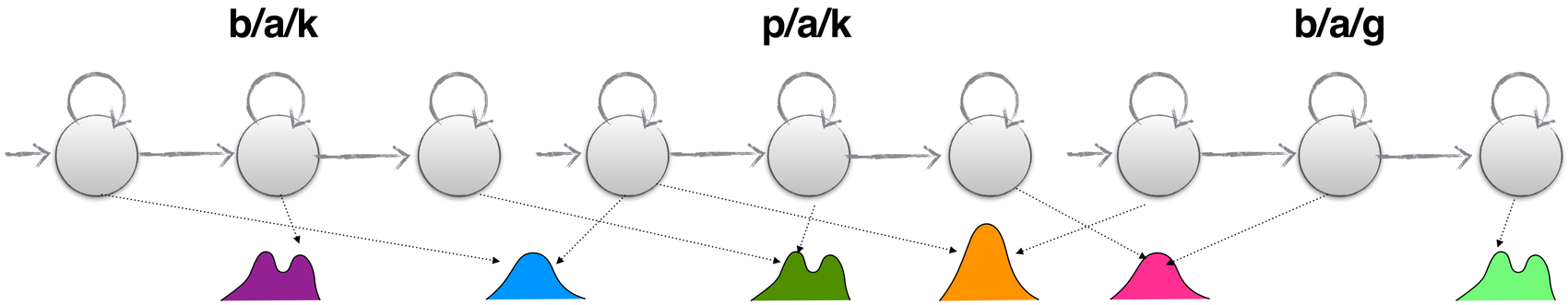
Triphone HMMs (Tied Mixture Models)

2. State Tying

- Observation probabilities are shared across states which generate acoustically similar data

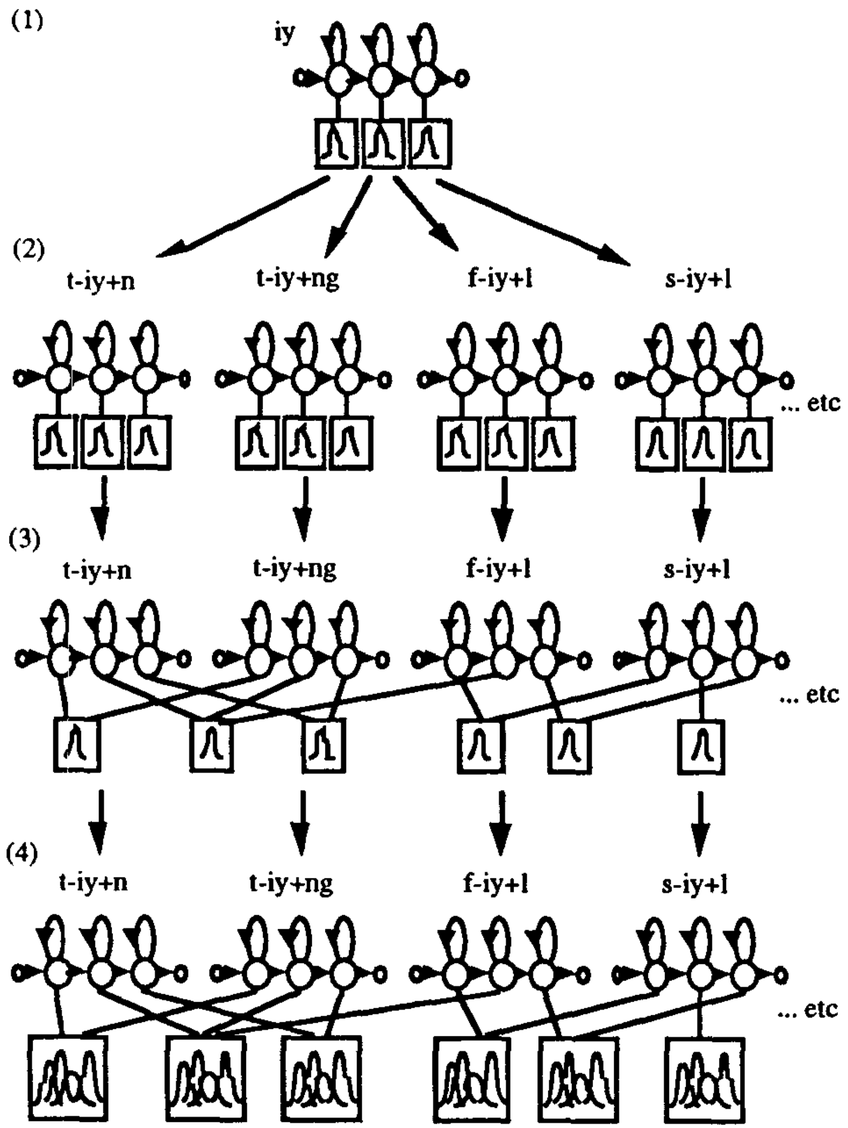


Triphone HMMs (No sharing)



Triphone HMMs (State Tying)

Tied state HMMs

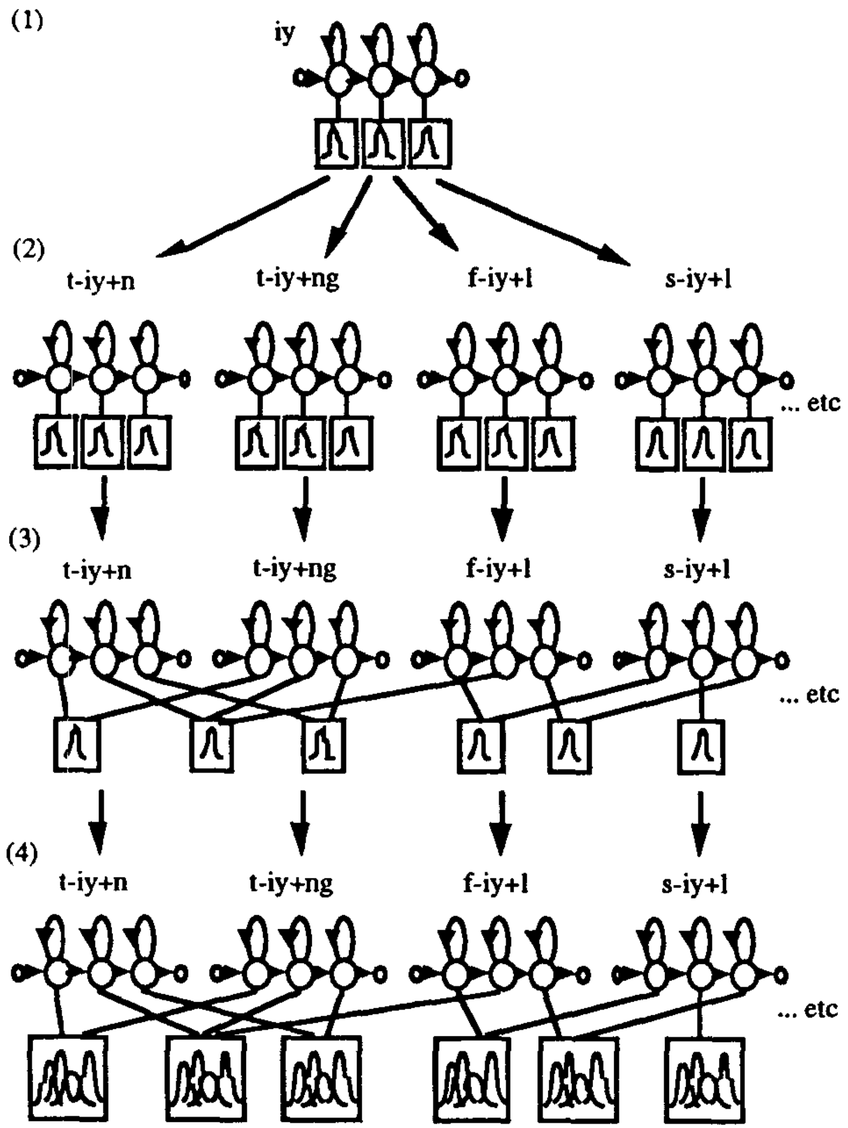


Four main steps in building a tied state HMM system:

1. Create and train 3-state monophone HMMs with single Gaussian observation probability densities
2. Clone these monophone distributions to initialise a set of untied triphone models. Train them using Baum-Welch estimation. Transition matrix remains common across all triphones of each phone.
3. For all triphones derived from the same monophone, cluster states whose parameters should be tied together.
4. Number of mixture components in each tied state is increased and models re-estimated using BW

Image from: Young et al., “Tree-based state tying for high accuracy acoustic modeling”, ACL-HLT, 1994

Tied state HMMs



Four main steps in building a tied state HMM system:

1. Create and train 3-state monophone HMMs with single Gaussian observation probability densities
2. Clone these monophone distributions to initialise a set of untied triphone models. Train them using Baum-Welch estimation. Transition matrix remains common across all triphones of each phone.
3. For all triphones derived from the same monophone, cluster states whose parameters should be tied toge
4. Num in each mod

Which states should be tied together? Use decision trees.

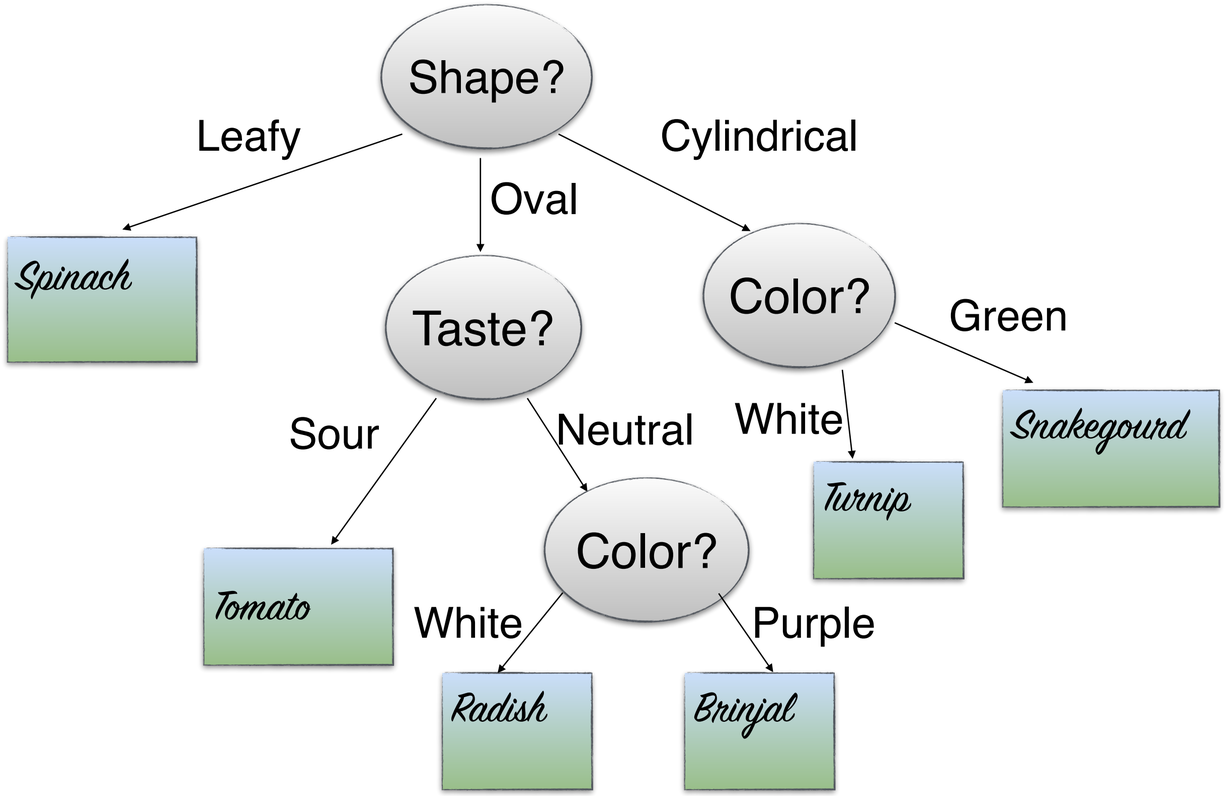
Image from: Young et al., "Tree-based state tying for high accuracy acoustic modeling", ACL-HLT, 1994

Decision Trees

Classification using a decision tree:

Begins at the root node: What property is satisfied?

Depending on answer, traverse to different branches

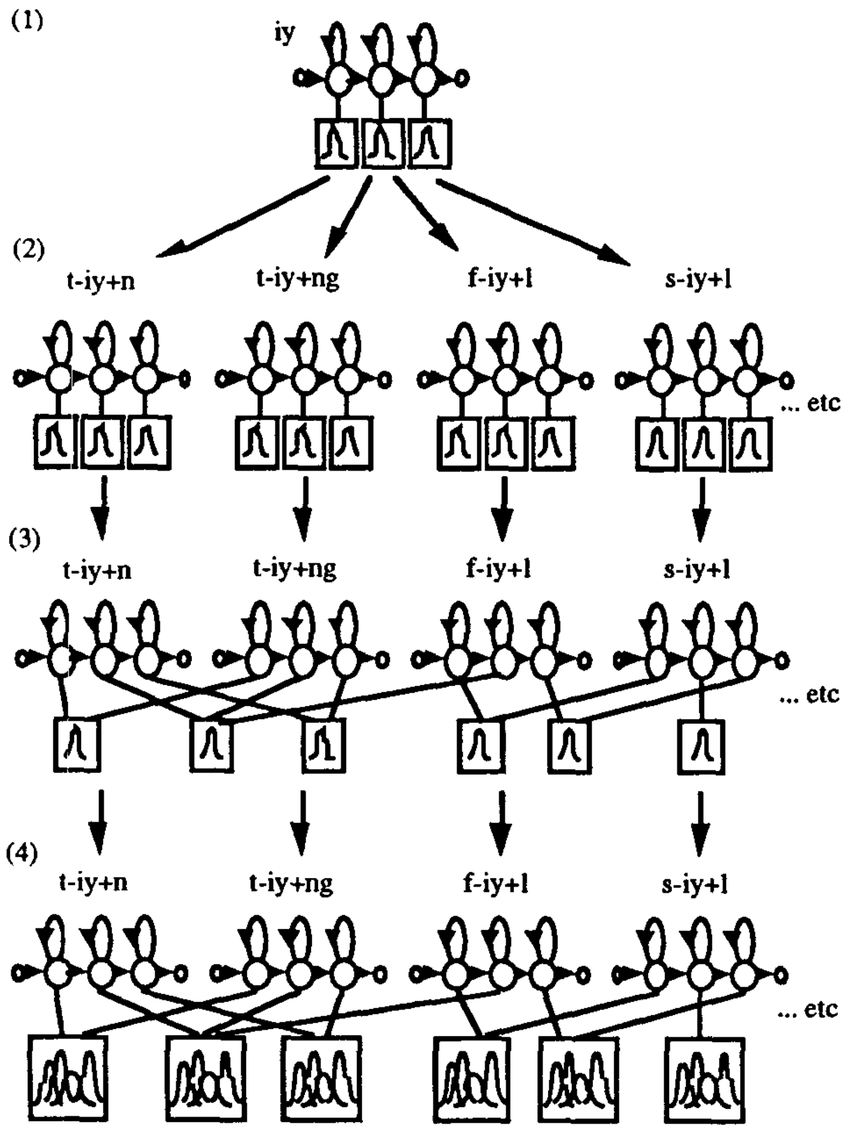


Decision Trees

- Given the data at a node, either declare the node to be a leaf or find another property to split the node into branches.

- Important questions to be addressed for DTs:
  1. How many splits at a node? Chosen by the user.
  2. Which property should be used at a node for splitting? One which decreases “impurity” of nodes as much as possible.
  3. When is a node a leaf? Set threshold in reduction in impurity

Tied state HMMs



Four main steps in building a tied state HMM system:

1. Create and train 3-state monophone HMMs with single Gaussian observation probability densities
2. Clone these monophone distributions to initialise a set of untied triphone models. Train them using Baum-Welch estimation. Transition matrix remains common across all triphones of each phone.
3. For all triphones derived from the same monophone, cluster states whose parameters should be tied toge
4. Num in each mod

Which states should be tied together? Use decision trees.

Image from: Young et al., "Tree-based state tying for high accuracy acoustic modeling", ACL-HLT, 1994

How do we build these phone DTs?

1. What questions are used?

   Linguistically-inspired binary questions: “Does the left or right phone come from a broad class of phones such as vowels, stops, etc.?” “Is the left or right phone [k] or [m]?”

How do we build these phone DTs?

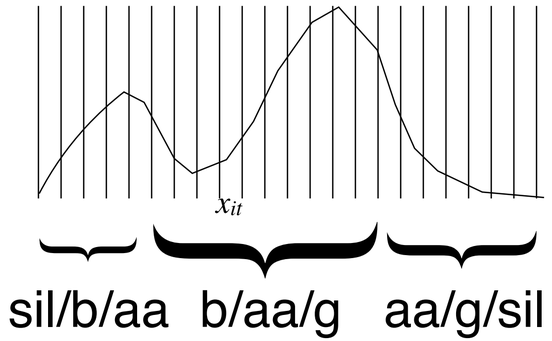
1. What questions are used?

   Linguistically-inspired binary questions: “Does the left or right phone come from a broad class of phones such as vowels, stops, etc.?” “Is the left or right phone [k] or [m]?”

2. What is the training data for each phone state, p_j? (root node of DT)

Training data for DT nodes

- Align training data, $x_i = (x_{i1}, \ldots, x_{iT_i})$ $i=1 \ldots N$ where $x_{it} \in \mathbb{R}^d$, against a set of triphone HMMs
- Use Viterbi algorithm to find the best HMM state sequence corresponding to each x_i
- Tag each x_{it} with ID of current phone along with left-context and right-context



x_{it} is tagged with ID aa_2[b/g] i.e. x_{it} is aligned with the second state of the 3-state HMM corresponding to the triphone b/aa/g

- For a state j in phone p, collect all x_{it}'s that are tagged with ID p_j[?/?]

How do we build these phone DTs?

1. What questions are used?

   Linguistically-inspired binary questions: “Does the left or right phone come from a broad class of phones such as vowels, stops, etc.?” “Is the left or right phone [k] or [m]?”

2. What is the training data for each phone state, p_j? (root node of DT)

   All speech frames that align with the j^{th} state of every triphone HMM that has p as the middle phone

3. What criterion is used at each node to find the best question to split the data on?

   Find the question which partitions the states in the parent node so as to give the maximum increase in log likelihood

Likelihood of a cluster of states

- If a cluster of HMM states, S = {s_1, s_2, …, s_M} consists of M states and a total of K acoustic observation vectors are associated with S, {x_1, x_2 …, x_K} , then the log likelihood associated with S is:

$$\mathcal{L}(S) = \sum_{i=1}^{K} \sum_{s \in S} \log \Pr(x_i; \mu_S, \Sigma_S) \gamma_s(x_i)$$

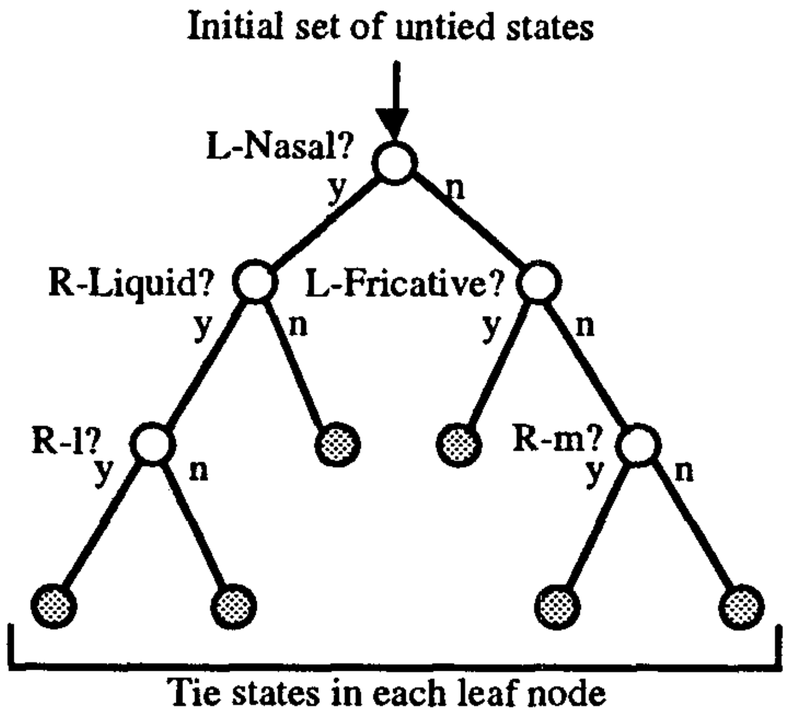
- For a question q that splits S into S_{yes} and S_{no}, compute the following quantity:

$$\Delta_q = \mathcal{L}(S^q_{\text{yes}}) + \mathcal{L}(S^q_{\text{no}}) - \mathcal{L}(S)$$

- Go through all questions, find Δ_q for each question q and choose the question for which Δ_q is the biggest

- Terminate when: Final Δ_q is below a threshold or data associated with a split falls below a threshold

Likelihood criterion



Given a phonetic question, let the initial set of untied states S be split into two partitions S_{yes} and S_{no}

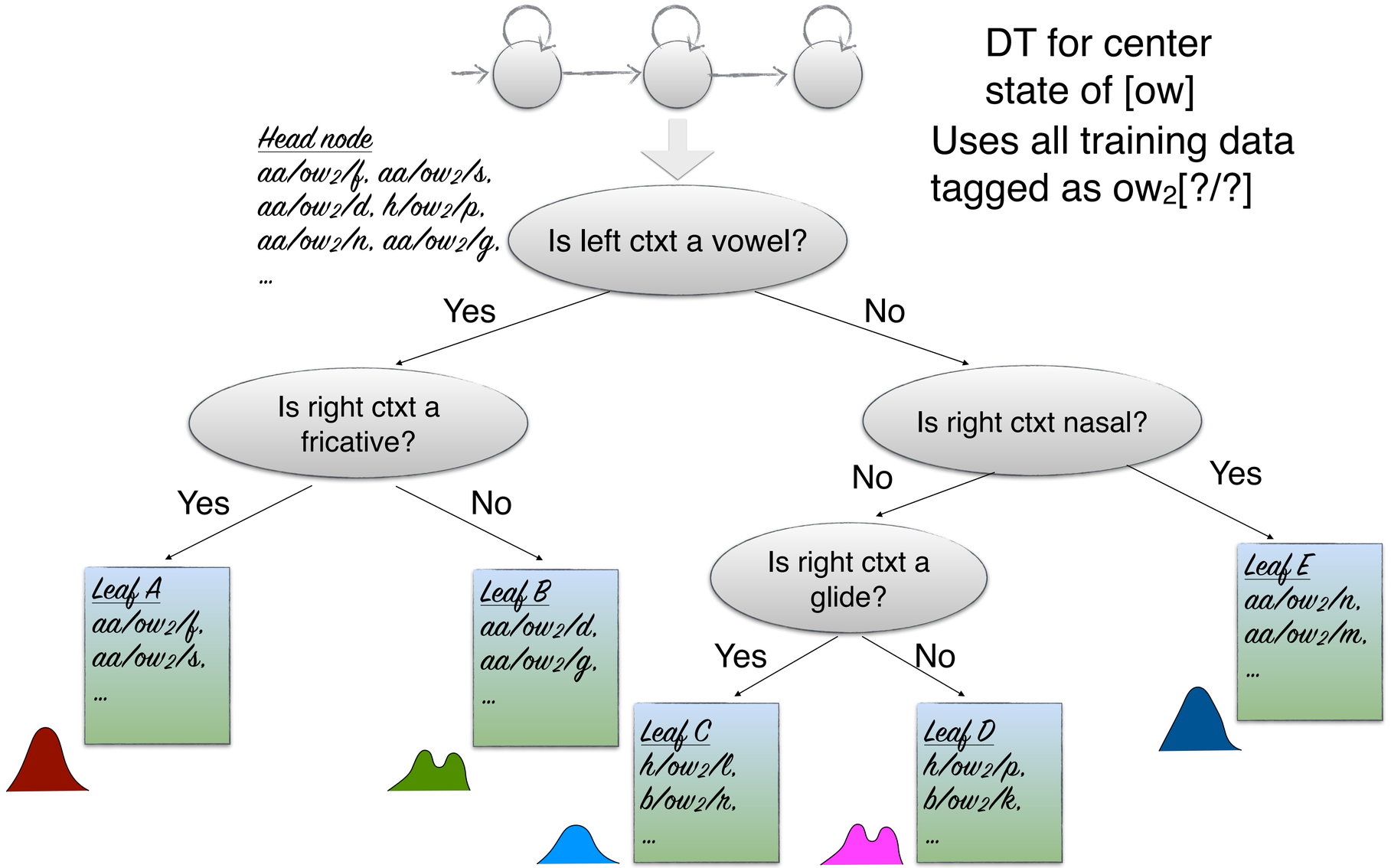
Each partition is clustered to form a single Gaussian output distribution with mean μ_{Syes} and covariance Σ_{Syes}

Use the likelihood of the parent state and the subsequent split states to determine which question a node should be split on

Image from: Young et al., “Tree-based state tying for high accuracy acoustic modeling”, ACL-HLT, 1994

Example: Phonetic Decision Tree (DT)

One tree is constructed for each state of each phone to cluster all the corresponding triphone states



DT for center state of [ow]

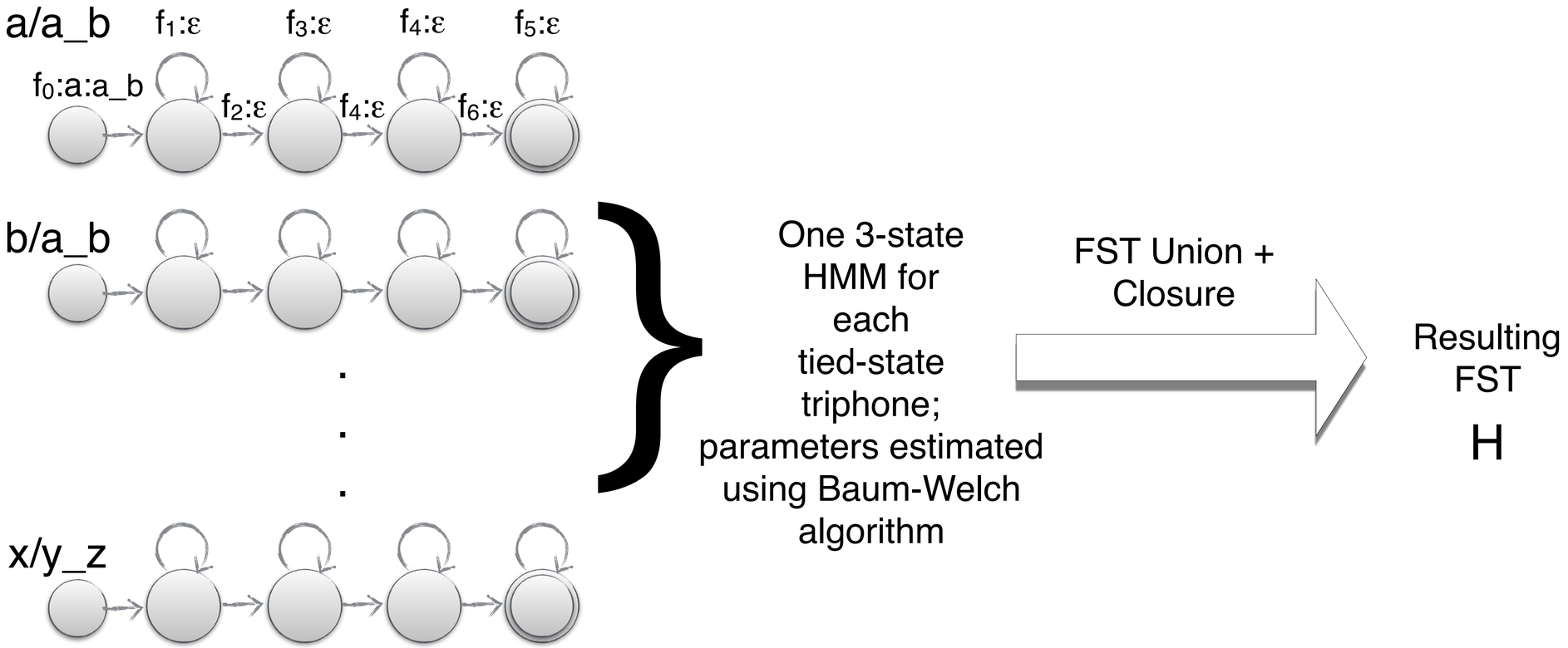
Uses all training data tagged as ow_2[?/?]

For an unseen triphone at test time

- Transition Matrix:
  - All triphones of a given phoneme use the same transition matrix common to all triphones of a phoneme
- State observation densities:
  - Use the triphone identity to traverse all the way to a leaf of the decision tree
  - Use the state observation probabilities associated with that leaf

That's a wrap on HMM-based acoustic models

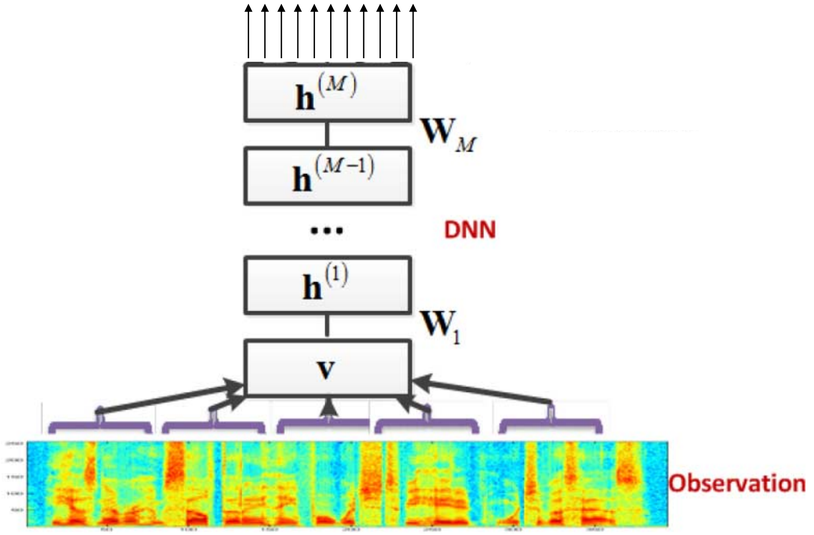
H



DNN-based acoustic models?

H

Phone posteriors



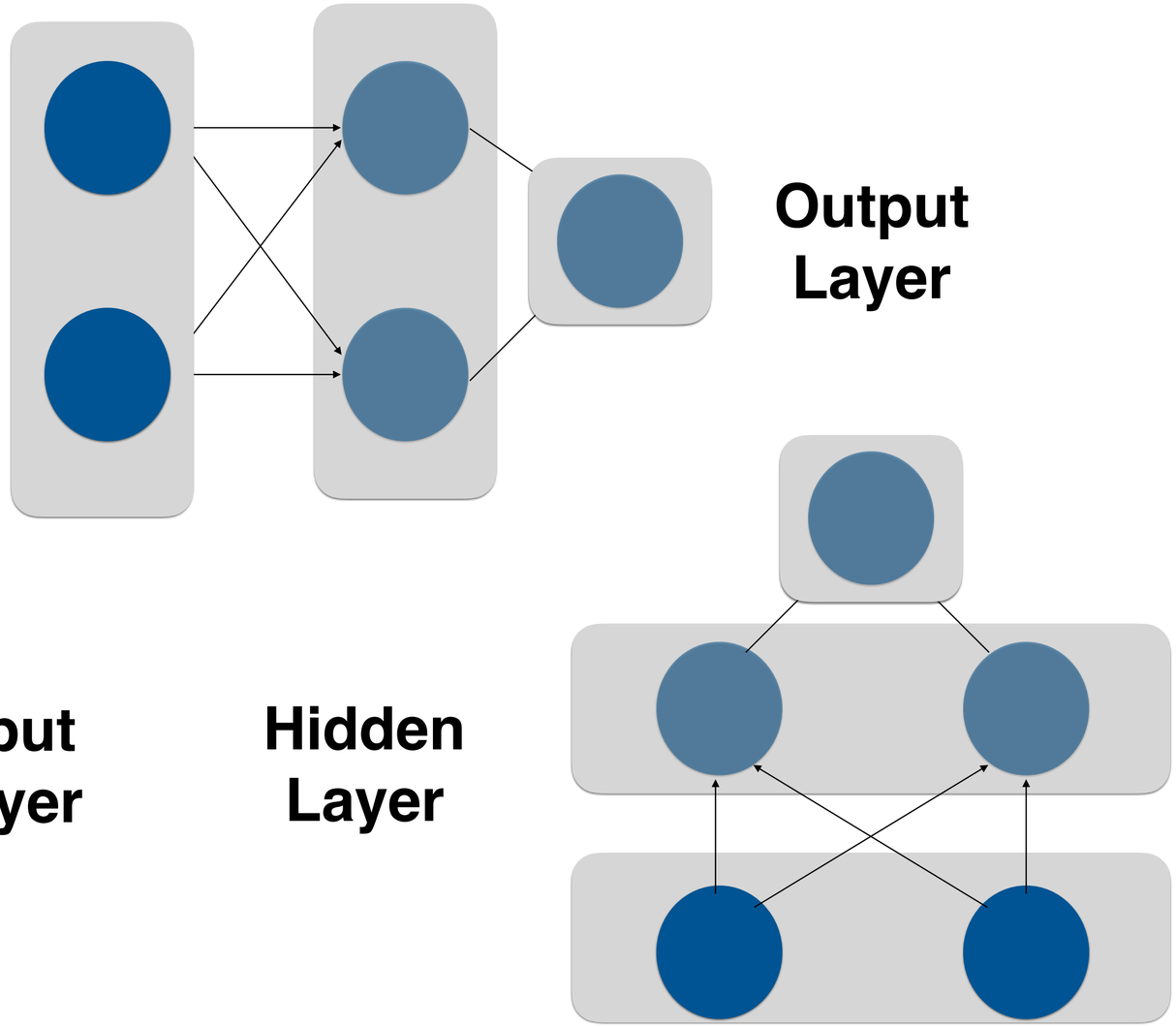
Can we use deep neural networks instead of HMMs to learn mappings between acoustics and phones?

Resulting FST

H

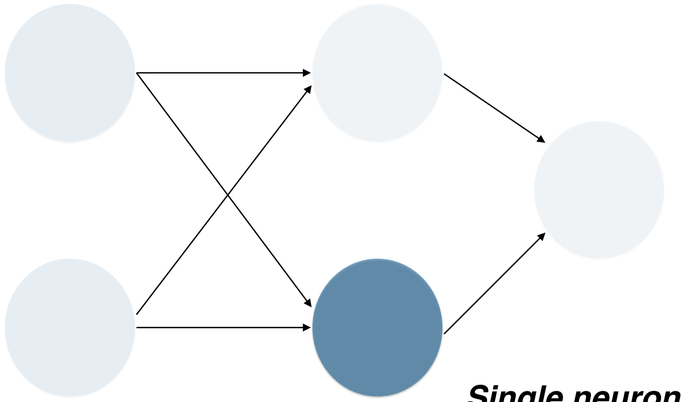
Brief Introduction to Neural Networks

Feed-forward Neural Network

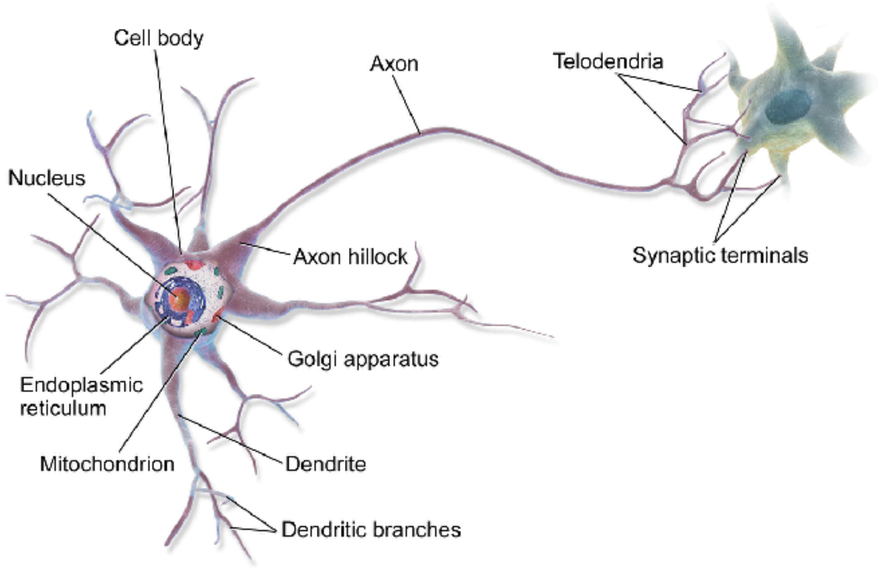


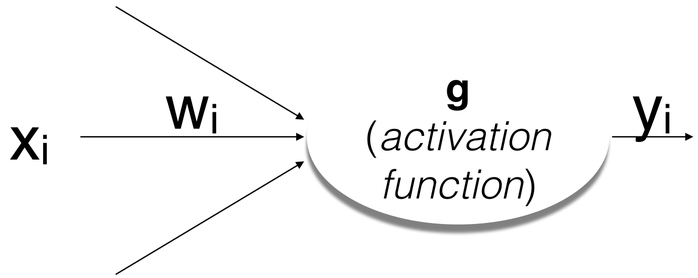
Feed-forward Neural Network

Brain Metaphor



Single neuron



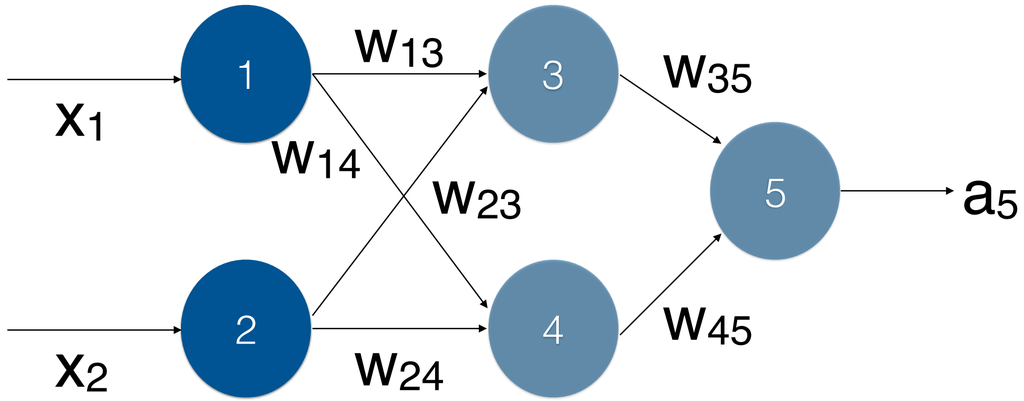


$$y_i = g(\Sigma_i w_i x_i)$$

Image from: https://upload.wikimedia.org/wikipedia/commons/1/10/Blausen_0657_MultipolarNeuron.png

Feed-forward Neural Network

Parameterized Model



$$a_5 = g(w_{35} \cdot a_3 + w_{45} \cdot a_4)$$

$$= g(w_{35} \cdot (g(w_{13} \cdot a_1 + w_{23} \cdot a_2)) + w_{45} \cdot (g(w_{14} \cdot a_1 + w_{24} \cdot a_2)))$$

Parameters of the network: all w_{ij} (and biases not shown here)

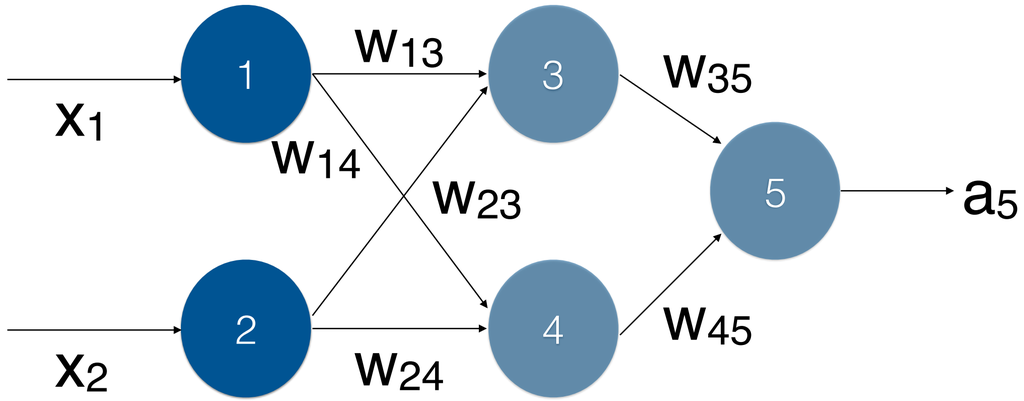
If x is a 2-dimensional vector and the layer above it is a 2-dimensional vector h, a fully-connected layer is associated with:

$$h = xW + b$$

where w_{ij} in W is the weight of the connection between i^{th} neuron in the input row and j^{th} neuron in the first hidden layer and b is the bias vector

Feed-forward Neural Network

Parameterized Model



$$a_5 = g(w_{35} \cdot a_3 + w_{45} \cdot a_4)$$

$$= g(w_{35} \cdot (g(w_{13} \cdot a_1 + w_{23} \cdot a_2)) + w_{45} \cdot (g(w_{14} \cdot a_1 + w_{24} \cdot a_2)))$$

The simplest neural network is the perceptron:

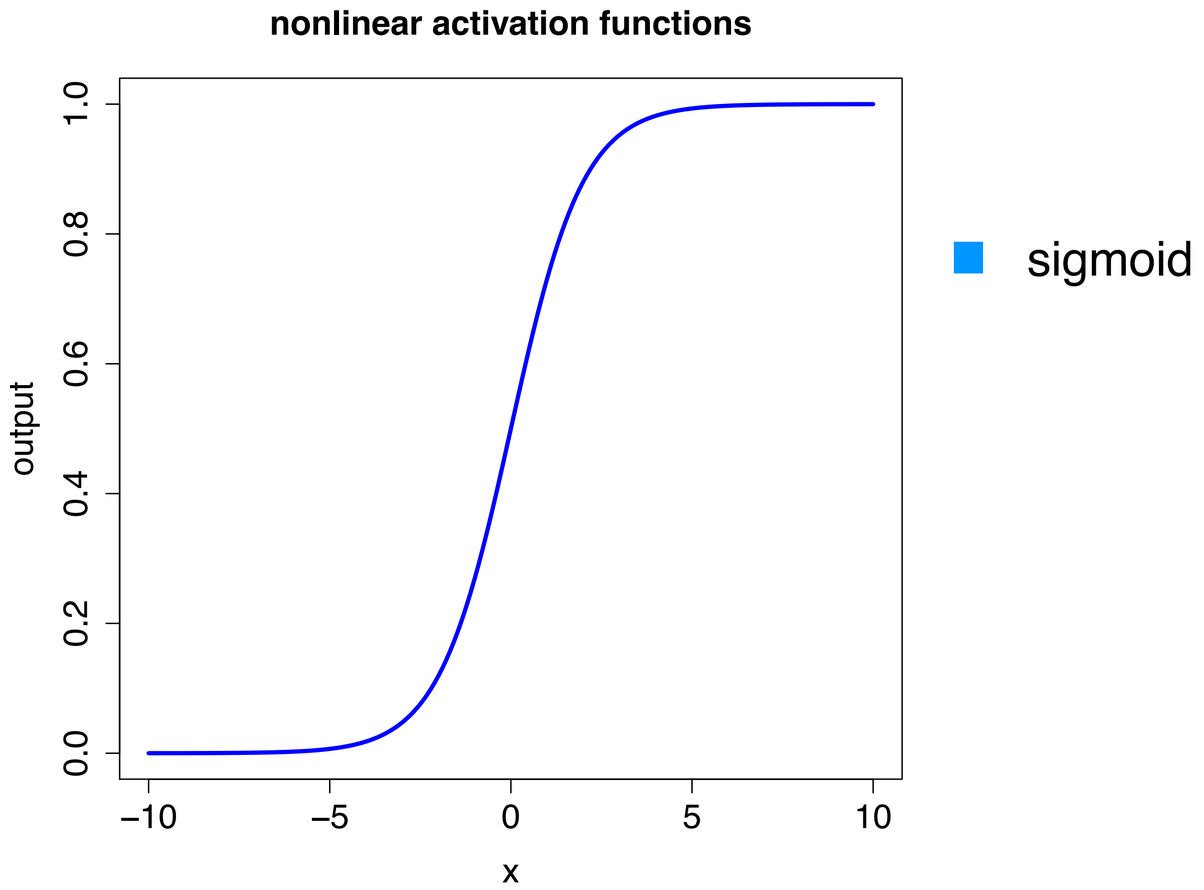
$$\text{Perceptron}(x) = xW + b$$

A 1-layer feedforward neural network has the form:

$$\text{MLP}(x) = g(xW_1 + b_1)\, W_2 + b_2$$

Common Activation Functions (g)

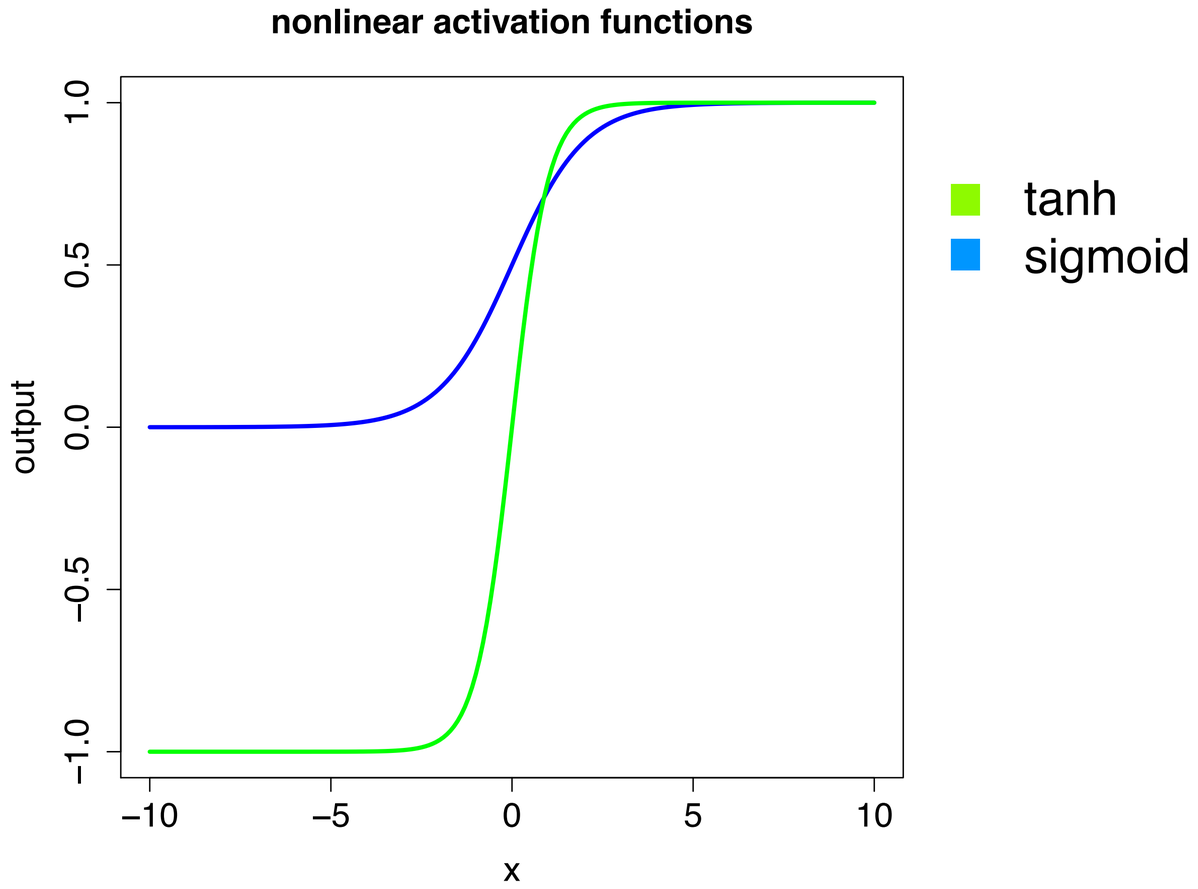
Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$



Common Activation Functions (g)

Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$

Hyperbolic tangent (tanh): $\tanh(x) = (e^{2x} - 1)/(e^{2x} + 1)$

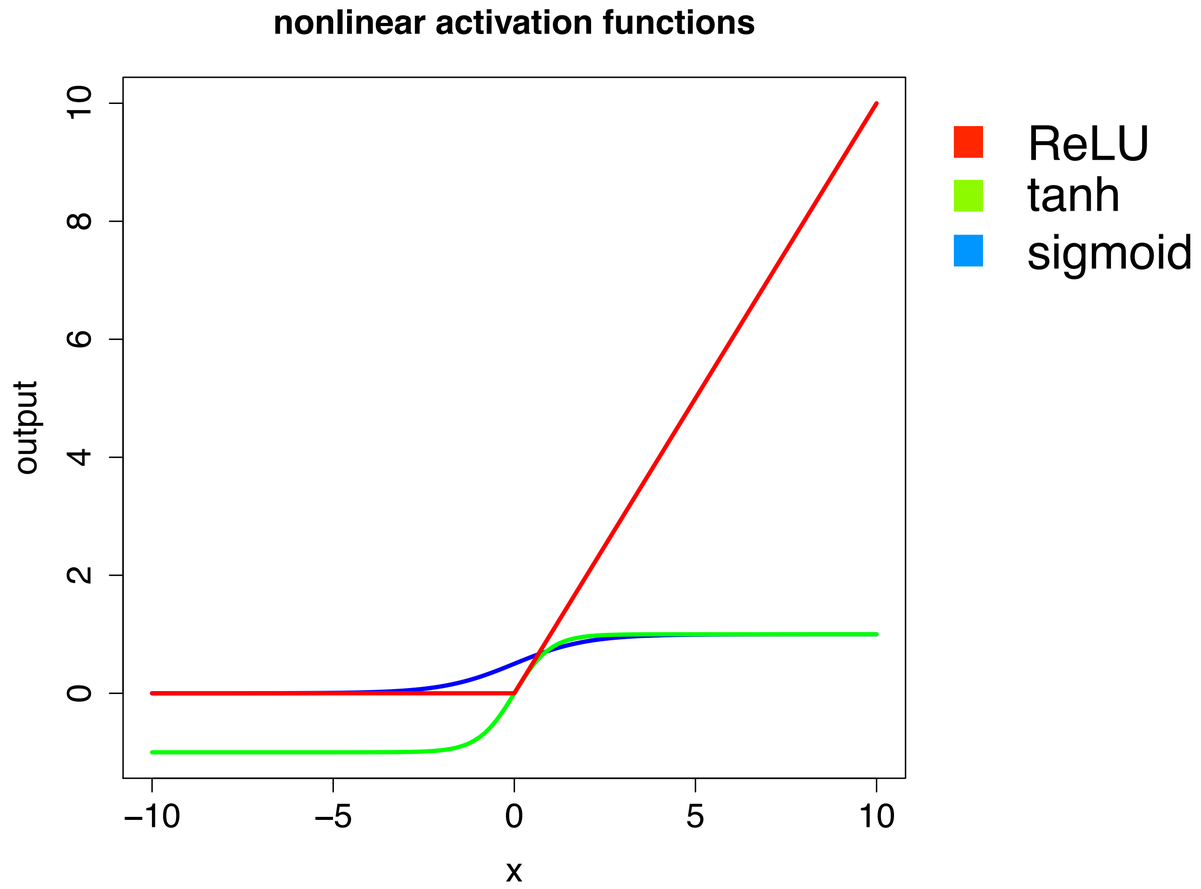


Common Activation Functions (g)

Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$

Hyperbolic tangent (tanh): $\tanh(x) = (e^{2x} - 1)/(e^{2x} + 1)$

Rectified Linear Unit (ReLU): $\text{RELU}(x) = \max(0, x)$



Optimization Problem

- To train a neural network, define a loss function $L(y,\tilde{y})$: a function of the true output y and the predicted output $\tilde{y}$
- $L(y,\tilde{y})$ assigns a non-negative numerical score to the neural network's output, $\tilde{y}$
- The parameters of the network are set to minimise L over the training examples (i.e. a sum of losses over different training samples)
- L is typically minimised using a gradient-based method

Stochastic Gradient Descent (SGD)

SGD Algorithm

Inputs:

Function NN(x; θ), Training examples, $x_1 \ldots x_n$ and outputs, $y_1 \ldots y_n$ and Loss function L.

do until stopping criterion

Pick a training example x_i, y_i

Compute the loss $L(NN(x_i; \theta), y_i)$

Compute gradient of L, ∇L with respect to θ

$\theta \leftarrow \theta - \eta \nabla L$

done

Return: θ

Training a Neural Network

Define the Loss function to be minimised as a node L

Goal: Learn weights for the neural network which minimise L

Gradient Descent: Find $\partial L/\partial w$ for every weight w, and update it as $w \leftarrow w - \eta\, \partial L/\partial w$

How do we efficiently compute $\partial L/\partial w$ for all w?

Will compute $\partial L/\partial u$ for every node u in the network!

$\partial L/\partial w = \partial L/\partial u \cdot \partial u/\partial w$ where u is the node which uses w

Training a Neural Network

New goal: compute $\partial L/\partial u$ for every node u in the network

Simple algorithm: Backpropagation

Key fact: Chain rule of differentiation

If L can be written as a function of variables $v_1, \ldots, v_n$, which in turn depend (partially) on another variable u, then

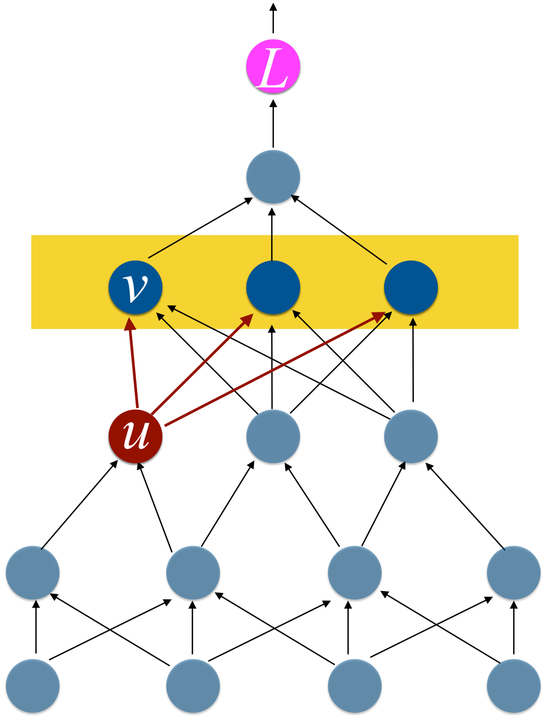
$$\partial L/\partial u = \Sigma_i \, \partial L/\partial v_i \cdot \partial v_i/\partial u$$

Backpropagation

If L can be written as a function of variables $v_1, \ldots, v_n$, which in turn depend (partially) on another variable u, then

$$\partial L/\partial u = \Sigma_i \, \partial L/\partial v_i \cdot \partial v_i/\partial u$$

Consider $v_1, \ldots, v_n$ as the layer above u, $\Gamma(u)$



Then, the chain rule gives

$$\partial L/\partial u = \Sigma_{v \in \Gamma(u)} \; \partial L/\partial v \cdot \partial v/\partial u$$

Backpropagation

$$\partial L/\partial u = \Sigma_{v \in \Gamma(u)} \; \partial L/\partial v \cdot \partial v/\partial u$$

<u>Backpropagation</u>

Base case: $\partial L/\partial L = 1$

For each u (top to bottom):

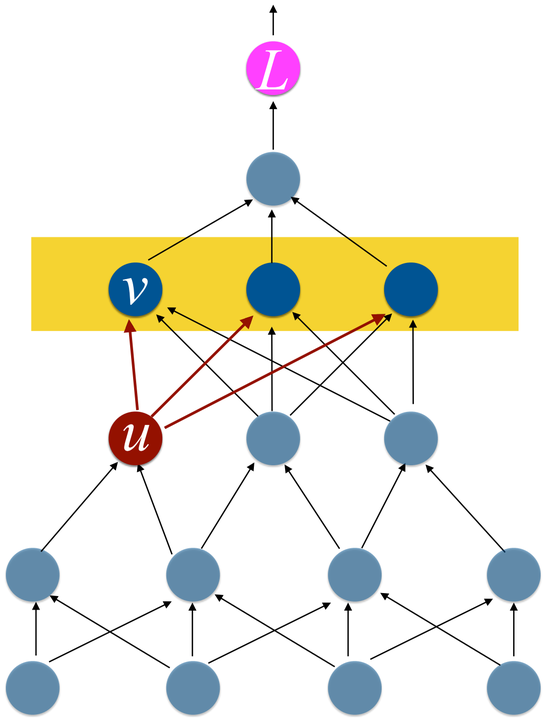
For each $v \in \Gamma(u)$:

Inductively, have computed $\partial L/\partial v$

Directly compute $\partial v/\partial u$

Compute $\partial L/\partial u$

Compute $\partial L/\partial w$
where $\partial L/\partial w = \partial L/\partial u \cdot \partial u/\partial w$



<u>Forward Pass</u>

First, in a forward pass, compute values of all nodes given an input
(The values of each node will be needed during backprop)

Where values computed in the forward pass may be needed

History of Neural Networks in ASR

- Neural networks for speech recognition were explored as early as 1987

- Deep neural networks for speech
  - Beat state-of-the-art on the TIMIT corpus [M09]
  - Significant improvements shown on large-vocabulary systems [D11]
  - Dominant ASR paradigm [H12]

[M09] A. Mohamed, G. Dahl, and G. Hinton, “Deep belief networks for phone recognition,” NIPS Workshop on Deep Learning for Speech Recognition, 2009.

[D11] G. Dahl, D. Yu, L. Deng, and A. Acero, “Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition,” TASL 20(1

[H12] G. Hinton, et al., “Deep Neural Networks for Acoustic Modeling in Speech Recognition”, IEEE Signal Processing Magazine, 2012.

What's new?

- Why have NN-based systems come back to prominence?
- Important developments
  - Vast quantities of data available for ASR training
  - Fast GPU-based training
  - Improvements in optimization/initialization techniques
  - Deeper networks enabled by fast training
  - Larger output spaces enabled by fast training and availability of data