# Introduction to Neural Networks

Steve Renals

Automatic Speech Recognition – ASR Lecture 10
23 February 2015

# Neural networks for speech recognition

- **Introduction to Neural Networks**
- **Training feed-forward networks**
- Hybrid neural network / HMM acoustic models
- Neural network features – Tandem, posteriorgrams
- Deep neural network acoustic models
- Neural network language models
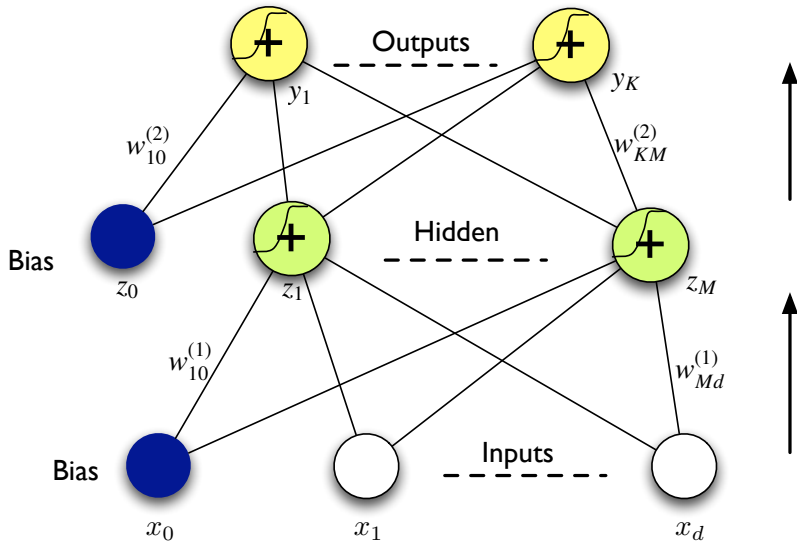
# Neural network acoustic models
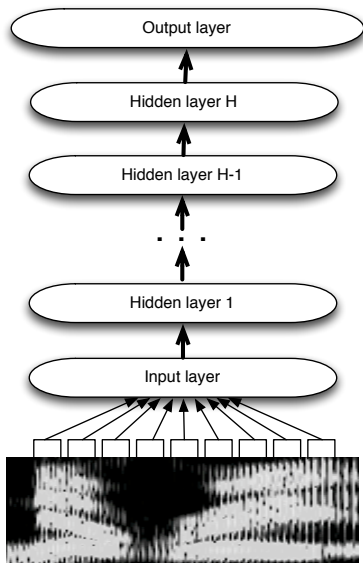


- Input layer takes several consecutive frames of acoustic features
- Output layer corresponds to classes (e.g. phones, HMM states)
- Multiple non-linear hidden layers between input and output
- Neural networks also called multi-layer perceptrons

# NN vs GMM
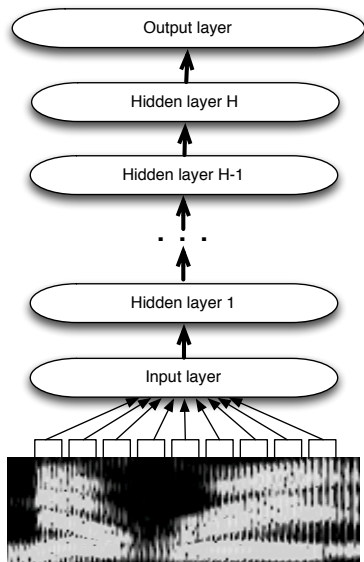


- Potential **deep** structure through multiple hidden layers (rather than a single layer of GMMs)
- Operates on multiple frames of input (rather than a single frame)
- One big network for everything (rather than one HMM per phone)

Neural network with one hidden layer

# Layered neural networks: structure (2)

- $d$ input units, $M$ hidden units, $K$ output units
- Hidden layer: each of $M$ units takes a linear combination of the inputs $x_i$:
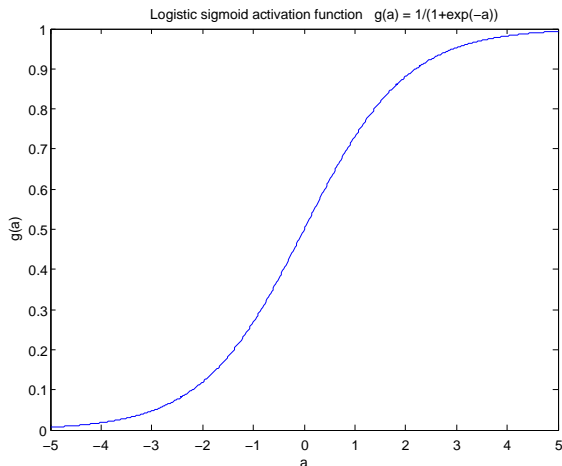
$$b_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i$$

  - $b_j$: activations
  - $w_{ji}^{(1)}$: first layer of weights

- Activations transformed by a nonlinear activation function $h$ (e.g. a sigmoid):

$$z_j = h(b_j) = \frac{1}{1 + \exp(-b_j)}$$

  - $z_j$: hidden unit outputs

# Logisitic sigmoid activation function

$$g(a) = \frac{1}{(1 + \exp(-a))}$$



Logistic sigmoid activation function   g(a) = 1/(1+exp(−a))

- Outputs of the hidden units are linearly combined to give activations of the output units:

$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

- Output units can be sigmoids, but it is better if they use the softmax activation function:

$$y_k = g(a_k) = \frac{\exp(a_k)}{\sum_{\ell=1}^{K} \exp(a_\ell)}$$

- Softmax enforces sum-to-one across outputs
- If output unit $k$ corresponds to class $C_k$, then interpret outputs of trained network as posterior probability estimates

$$P(C_k \mid \mathbf{x}) = y_k$$

# Layered neural networks: training

- Weights $w_{ji}$ are the trainable parameters
- Train the weights by adjusting them to minimise a cost function which measures the error the network outputs $y_k^n$ (for the $n$th frame) compared with the *target* output $t_k^n$
  - Sum squared error

  $$E^n = \frac{1}{2} \sum_{k=1}^{K} ||t_k^n - y_k^n||^2$$

  - For multiclass classification, the natural cost function is (negative) log probability of the correct class

  $$E^n = - \sum_{k=1}^{K} t_k^n \log y_k^n$$

- $E = \sum_n E^n$
- Optimise the cost function using *gradient descent* (back-propagation of error — backprop)

# Gradient descent

- Gradient descent can be used whenever it is possible to compute the derivatives of the error function $E$ with respect to the parameters to be optimized $\mathbf{W}$
- Basic idea: adjust the weights to move downhill in *weight space*
- Weight space: space defined by all the trainable parameters (weights)
- Operation of gradient descent:
  1. Start with a guess for the weight matrix $\mathbf{W}$ (small random numbers)
  2. Update the weights by adjusting the weight matrix in the direction of $-\nabla_{\mathbf{W}} E$.
  3. Recompute the error, and iterate
- The update for weight $w_{ki}$ at iteration $\tau + 1$ is:

$$w_{ki}^{\tau+1} = w_{ki}^{\tau} - \eta \frac{\partial E}{\partial w_{ki}}$$

The parameter $\eta$ is the *learning rate*

# Neural Networks vs GMMs

1. **Better use of training data**. Each Gaussian component of a GMM is mainly dependent on just a small subset of the data (divide and conquer); Each hidden unit of an NN is constrained by a large part of the data.

2. **Modelling multiple events**. NNs can model multiple events in the input simultaneously – different sets of hidden units modelling each event; GMMs assume each data point is generated by a single mixture component.

3. **Input context**. NNs can exploit multiple (correlated) frames of input; GMMs with diagonal covariance matrics require decorrelated inputs

4. **Parallelisation**. EM algorithm used to train GMMs is much easier to parallelise than stochastic gradient descent.

# End of part 1

- Next two lectures
    - Hybrid NN/HMM systems
    - Tandem features
    - Deep neural networks
    - Neural network language models
- Reading
    - N Morgan and H Bourlard (May 1995). Continuous speech recognition: An introduction to the hybrid HMM/connectionist approach, IEEE Signal Processing Magazine, 12(3), 24-42.
    - Christopher M Bishop (2006). *Pattern Recognition and Machine Learning*, Springer. Chapters 4 and 5.