NEURAL NETWORKS AND PRONUNCIATIONS

Neeraj Deshmukh

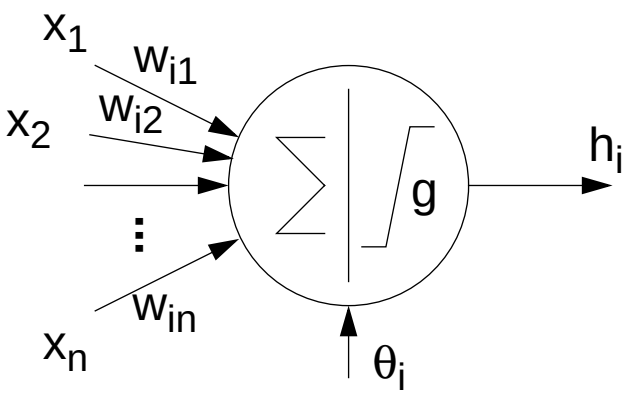
Institute for Signal & Information Processing
Mississippi State University
Mississippi State, MS 39762
deshmukh@isip.msstate.edu

ISIP Weekly Seminar Series
Fall 1997
August 28, 1997

INTRODUCTION

- Pronunciation modeling problem
  - voice-based interfaces
  - accurate recognition & synthesis
  - proper nouns — letter-to-sound rules do not apply
  - multiple pronunciations
- Why neural network classifiers
  - nonlinear dynamics
  - ability to generalize
  - other techniques
    - — rule-based systems do not work
    - — decision trees

BASICS OF A NEURAL NETWORK



The i^{th} neuron

- Terminology
  - inputs — x_j
  - connecting weights — w_{ij}
  - threshold term — θ_i
  - output — h_i
  - activation function — g()

- Activation

$$h_i(t+1) = g(\sum_j w_{ij} x_j(t) - \theta_i)$$

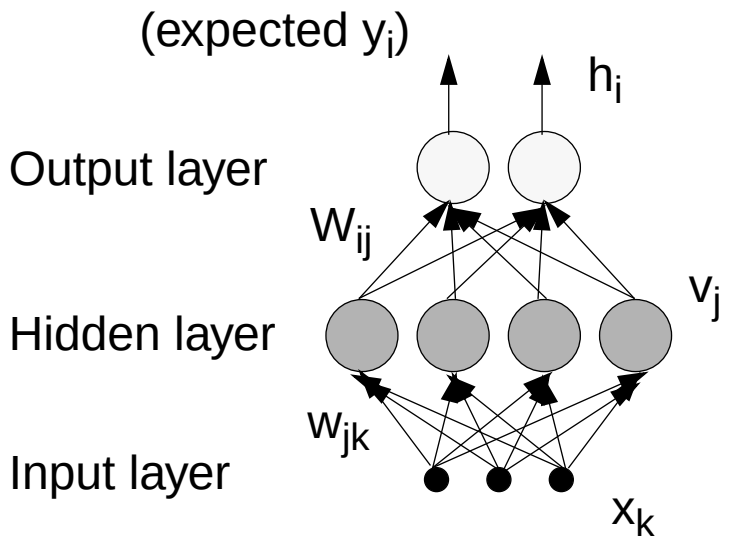
- Hebb rule

$$w_{ij} \propto x_i x_j \qquad \text{or} \qquad w_{ij} = \frac{1}{N} x_i x_j$$

TYPICAL ARCHITECTURES

- Hopfield Networks
  - generalized Hebbian
  - symmetric
- Perceptrons
  - Feedforward networks (Multilayered perceptrons)
- Learning Vector Quantizers
- Stochastic Networks
  - Boltzmann machine
- Radial Basis Functions
- Time-delayed Neural Networks

MULTILAYERED PERCEPTRONS



A two-layered MLP

- Feedforward networks
- Hidden layer(s) capture the distribution of the training data
- Activation function determines ability to classify
  - g() = sgn — can classify linearly separable data
  - g() linear — can classify linearly independent data

- Backpropagation training — a variant of gradient descent

BACKPROPAGATION LEARNING

- Update weights in proportion with the error gradient

$$\Delta W = -\eta \frac{\partial}{\partial W} \varepsilon(W) \qquad \text{where} \qquad \varepsilon(W) = \frac{1}{2} \sum_i (y_i - h_i)^2$$

- Derivation of training algorithm
  - We have

$$v_j = g(\sum_k w_{jk} x_k)$$

$$h_i = g(\sum_j W_{ij} v_j) = g(\sum_j W_{ij} g(\sum_k w_{jk} x_k))$$

  - Error becomes

$$\varepsilon = \frac{1}{2} \sum_i (y_i - h_i)^2$$

$$= \frac{1}{2} \sum_i \left(y_i - g(\sum_j W_{ij} g(\sum_k w_{jk} x_k)) \right)^2$$

BACKPROPAGATION (CONTD.)

- Weight update terms
  - for output layer

$$\Delta W_{ij} = -\eta \frac{\partial \varepsilon}{\partial W_{ij}} = \eta \xi_i v_j$$

$$\text{where} \quad \xi_i = g'(\sum_j W_{ij} v_j)(y_i - h_i)$$

  - for hidden layer

$$\Delta w_{jk} = -\eta \frac{\partial \varepsilon}{\partial w_{jk}} = \eta \delta_j x_k$$

$$\text{where} \quad \delta_j = g'(\sum_k w_{jk} x_k) \sum_i W_{ij} \xi_i$$

- Easily extensible to multiple hidden layers

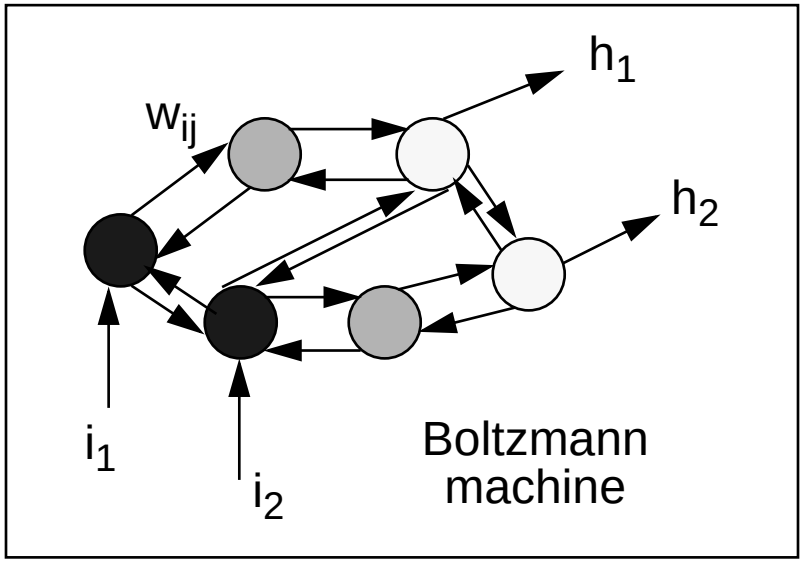
BACKPROPAGATION (SUMMARY)

- Forward pass
  - set the inputs to the network
  - compute outputs of successive hidden layers
  - compute output of final (output) layer
  - compare with desired output and compute error function
- Backward pass
  - compute error terms for output layer ξ_i
  - update the output layer weights using $W_{ij} = W_{ij} + \Delta W_{ij}$
  - backpropagate error to hidden layer $\sum_i W_{ij}\xi_i$
  - update the hidden layer weights using $w_{jk} = w_{jk} + \Delta w_{jk}$

PROPERTIES OF MULTILAYERED PERCEPTRONS

- Very good classifiers for linearly independent data
- Can also solve nonlinear classification problems with appropriate activation functions
- Good for generalization and pattern recognition
- Require supervised learning
- Training is computationally intensive
- No clear idea about the correct number of hidden layers and hidden neurons
- Most popular class of neural network classifiers

BOLTZMANN MACHINE



- Stochastic Hopfield network
- Energy function

$$E\{\underline{x}\} = -\frac{1}{2}\sum_i \sum_j w_{ij} x_i x_j$$

decreases monotonically as the system evolves according to its dynamical rule

- Activation is probabilistic

$$p(x_i) = g(\sum_j w_{ij} x_j) = \left(1 + e^{-\frac{1}{T}\sum_j w_{ij} x_j} \right)^{-1}$$

BOLTZMANN DISTRIBUTION

- For i^{th} neuron, the energy difference between ON and OFF states is

$$\Delta E_i = \sum_j w_{ij} x_j$$

- At a given temperature T with thermal equilibrium
  - i^{th} neuron ON — global state α
  - i^{th} neuron OFF — global state β

  Then the relative probability of the two states follows the Boltzmann-Gibbs distribution

$$\frac{P_\alpha}{P_\beta} = e^{-\frac{E_\alpha - E_\beta}{T}}$$

SIMULATED ANNEALING

- Set system temperature
- Set external inputs
- Randomly activate neurons based on energy gap and temperature
- Update weights such that

$$\Delta w_{ij} \propto [x_i, x_j]_{Data} - [x_i, x_j]_{P(\underline{x}/\underline{W})}$$

where

$[x_i, x_j]_{Data}$ = Correlation based on data

$[x_i, x_j]_{P(\underline{x}/\underline{W})}$ = Correlation based on the current distribution

- Reduce temperature and repeat

PROPERTIES OF BOLTZMANN MACHINES

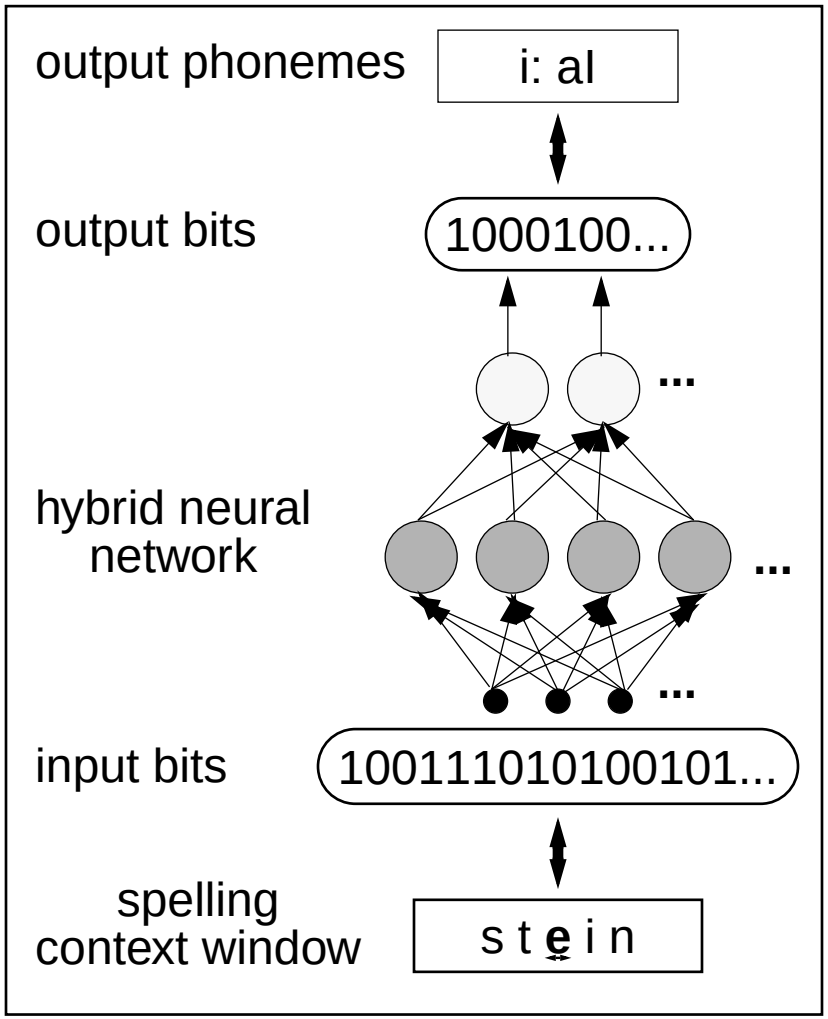
- Powerful statistical models
- Can accurately capture up to second-order statistics of data
- Can solve nonlinear problems such as the Traveling Salesman
- Requires Monte Carlo methods to estimate correlation terms in training
- Computationally very expensive to train
- Not very popular

SO HOW DO PRONUNCIATIONS FIT IN?

- Hybrid network — combine elements of
  - Multilayered perceptrons
  - Boltzmann machines
- System components
  - Input — binary-encoded letters (spelling) of the word
  - Output — phonemes that form the pronunciation
  - Training data — a word-pronunciation aligned dictionary
- Hidden neurons capture statistics of sounds w.r.t. letters in context of surrounding letters

  e.g. ‘ur’ by itself is *3r* (cur), but ‘ure’ is *j u &r* (cure)
- Probabilistic activation of neurons provides multiple pronunciations

HYBRID NETWORK FOR PRONUNCIATIONS



- Topology
  - Multilayered perceptron
  - Stochastic activation of neurons
  - Activation function

$$g(\,) = \tanh\left(\frac{1}{T}\sum_{j} w_{ij} x_j\right)$$

  - Each output layer neuron corresponds to a phone
  - Active output neurons can be rank-ordered to get a N-best list

TRAINING ALGORITHM

- Backpropagation with simulated annealing
  - In forward pass
    - temperature term controls activation of neurons
    - activation rule is stochastic

$$p(v_j = 1) = \tanh\left(\frac{1}{T}\sum_k w_{jk}x_k\right)$$

$$p(h_i = 1) = \tanh\left(\frac{1}{T}\sum_j W_{ij}v_j\right)$$

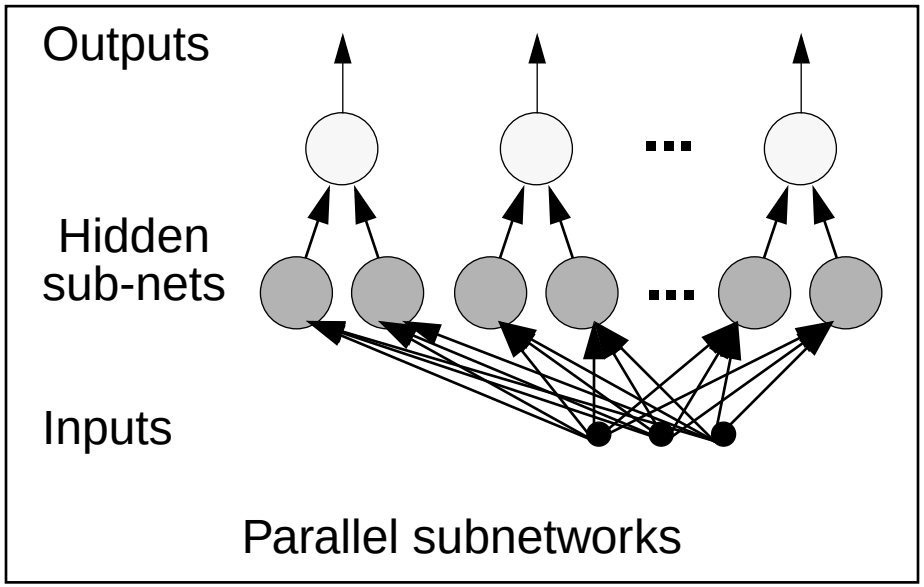
  - In backward pass
    - weight updates follow the backpropagation rules
  - Temperature is reduced in successive iterations

PRONUNCIATIONS DICTIONARY

- 18494 proper nouns (surnames)
  - mostly American and European names, some Oriental
  - 73 alphanumeric characters including punctuation marks
- 24041 pronunciations
  - hand-transcribed
  - automatic letter-phone alignment — ‘blank’ phoneme
  - Modified Worldbet set — 46 phonemes

Einstein	_ aI n s t _ aI n _ aI n s t _ i: n
Elmsford	E _ m s f _ >r d

EXPERIMENTS



- Baseline system
- Parallel subnetworks
  - Each phoneme has its own subnetwork
  - Sub-net size determined by the *a priori* phone likelihood

- Exponentially decaying temperature —

$$T_n = T_{n-1} e^{-\tau}$$

- Training data — 15000 names
- Test set — (held-out) 3494 names

RESULTS

- Test 1 — data subset for 6 phonemes

System	context length	# hidden neurons	# iterations	N-best	% phones correct	
					Training	Test
Baseline	5	128	20	3	100.00	100.00
		72	18		100.00	99.92
Parallel Sub-nets		256	32		100.00	99.92
		128	27		99.76	98.48

- Test 2 — full data set for all 46 phonemes

System	context length	# hidden neurons	# iterations	N-best	% phones correct	
					Training	Test
Baseline	5	2048	60	3	73.12	66.28
		1024	100		62.71	58.33
Sub-nets		1024	60		61.94	56.56

CONCLUSIONS

- Neural networks for multiple pronunciation generation
- Hybrid network works well on small classification problems
- For the parameters tested, the network fails to scale up
  - insufficient training
  - highly confusible data
  - sparse data for rare phonemes

    — e.g. occurrence of *@* is 0.037466 while that of *iU* is 0.000170
- Future directions
  - Multiple hidden layers
  - Integration with decision trees