Neural Networks Ch. 19.4-19.5

1 Basic Assumptions

- Model Network (inspiration)
- Representation
- input: attributes with real-values
  - boolean values treated as {0,1} or {-1,1}
  - typical .9 is treated as 1, .1 as 0
- prediction: real or symbolic attribute
- Competitive algorithm, i.e.

2 Algorithm: Back Propagation

FeedForward network: (no recurrent arcs, unlike people)

- each node is usually an LTU (binary or real output)
- each layer passes output to next layer
- Nodes of Intermediate layers called Hidden Units
- Idea of Learning Algorithm: pass error signal backwards, sharing responsible in proportion to weight.

3 Properties

- incremental
- failure driven
- noise tolerant
- constructive induction via hidden units
- SLOW to learn but executes quickly
- supports typicality (fuzzy decisions)
- natural support of real-values
- unnatural support of symbols
- 2-layer network is Boolean (representationally) complete: why?

4 Internal Nodes

LTU This is the standard model. Defined by:
Fire if: $\sum_{i=1}^{i=n+1} I_i * W_i > 0$
This yields a sharp cliff.

Sigmoid This is a differential approximation of the standard model, which allows the derivation of Backpropagation. Output of node is logistic function, i.e. $\frac{1}{1+e^{-x}}$ where x is the $I \cdot W$.
This yields smooth cliffs.
Note that $\lim_{x\to\infty} logistic(x) = 1$
and $lim_{x\to-\infty} logistic(x) = 0$.

Gaussian This is also differentiable, and yields radial basis networks. The same derivation method can be used to yield update rules. Here the output of a node is defined by $e^{-2/\sigma\sum_i (x_i - c_i)^2}$. This has a maximum value of 1 at $x_i = c_i$ and goes to 0 as x moves away from c.

LTU and Sigmoid functions make positive and negative predictions over the whole range of inputs. Gaussian nodes only make local positive prediction.

5 Idea of BackProp

- Goal: Make the net a differentiable function
- Method: Replace threshold nodes by differentiable approximation
- Sigmoid functions is one way.
- Adjust weights to minimize Error function, now differentiable.
- Method: Use gradient descent in weight space
- Method: Take partial derivatives wrt weights
- for Hidden nodes, use chain rule

6 Derivation

As with perceptron, we can define error signal via Err =(T-0).

We try to minimize $Err = 1/2 * \sum_i (T_i - O_i)^2$.

Again we take partial derivatives with respect to the weight function. That's why we choose the logistic function $\sigma(x)$. The derivate of it is $\sigma(x)(1 - \sigma(x))$ which can be approximated by $x(1-x)$. The whole computation is painful but essentially involves computing the partial derivates with respect to the parameters, and this tells us how to modify the weights.

The net effect is that backpropagation carries out a steepest descent change on surface whose height is the error measure, ie the square of the difference between the desired output and the actual output. This is similar to LTU learning rule. No change if right, change in proportion to error and learning rate if wrong.

This can be repeated to generate a rule for updating Gaussian networks.

7 BackPropagation Update rules

This is just for the 2-layered case, but similar results hold in general.

Weight Update Rules:

Definitions:

Regard the neural net as function from R^N to R^M.
O_i is the actual ith output
T_i is the desired ith output (teacher signal)
Err_i is the error $(T_i - O_i)$
$W_{j,i}$ is weight on arc from jth node to ith node. (beware of difference between books)
α is learning rate, >0, often .1

a_j is the activation of the Hidden Unit a_j.
$g'(in_i)$ is the derivative of the activation function g.

For Output layer:

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times Err_i \times g'(in_i)$$

If Δ_i is $Err_i \times g'(in_i)$ then

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

For Hidden Units:

$\Delta_j = g'(in_i) \sum_i W P_{j,i} \Delta_i$

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$$

8 Performance

Network	Rate	Or	And	Xor
2HU	1.0	297 (3/10)	301 (3/10)	394 (3/10)
2HU	2.0	345 (5/10)	152 (6/10	199 (6/10)
1HU	1.0	5 (10/10)	12 (10/10)	264 (9/10)
1HU	2.0	5 (10/10)	6 (10/10)	145 (10/10)

k(n/p) means p is the number of initial weights, n is the number of trials that were successful (after at most 1000 weight changes) and k is the average number of weight changes when successful.

There are 2 inputs, 1 or 2 HU, and 1 output.

9 Extensions

- Recurrent networks
- information about state, e.g. alpha if left guy alpha.
- radial basis functions (need to change back-prop)
- k-means clustering for new nodes
- over same class or different classes
- Add momemtum term:

$$\delta w_{ij} = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ji}(n)$$

- sigma-pi nodes (use product rather than sum)
- Structure of the net
  - KBann - initialize net via a theory (propositional)
  - Cascade: add nodes as needed, one at time
  - Shavlik- uses fsa to initialize
- extract rules from network
- Hidden units for data compression: ID to Hidden to ID. Or code learning.

10 Performance

- Usually has boolean completeness property,
- Very good on perceptual problems
- Sometimes best performance
- Available
- Thousands of sweeps or epochs needed for XOR, and occassional network doesn't learn concept.
- standard test functions: parity, multiplexor

11 Successes

Nettalk Learned to pronounce words from 1024 examples.
Network structure: 29 input units, 80 hidden units, ? output units.
On training data achieves 95% accuracy and on test data 78% accuracy.
Worse than hand-coded but still impressive.
Learning curves (via vocalization) sounds like children learning lanuage.

Disease diagnosis Journal of Urology, November, 1994
90% accurate at predicting prostate cancer

Handwritten Character Recognition 5-layered net:
256(16 by 16)-768-190-30-10(the digits)
Not completely connected. After training on 7300 examples, achieved 99% accuracy.
Learned network implement in VLSI for speed.

Driving Alvinn is an autonomous vehicle developed at CMU.
Input:30X32. Output 30 units corresponding to steering directions.
5 HUs fully connected to input and output.
Training done via humans experts plus deviations.
5 minutes of training data sufficient for a road conditions.
No theory of driving but data easy to get.

12 Problems

- structure of net
- representation of symbols, variables, relations
- stuck in local maximum
- comprehensibility
- slowness of learning
- invariance: in learning T's and C's in different orientations results in construction of specialized HU's.

13 Questions

- How would you do XOR as a neural net?
- Can you represent exactly 5 of 10 as neural net?
- Suppose a concept is linearly separable. Can a neural net learn it? Will it do a better job than a perceptron?