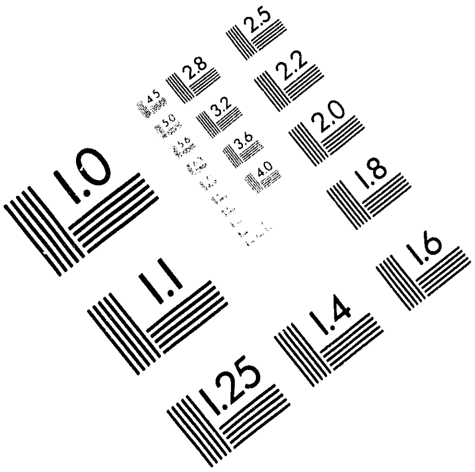
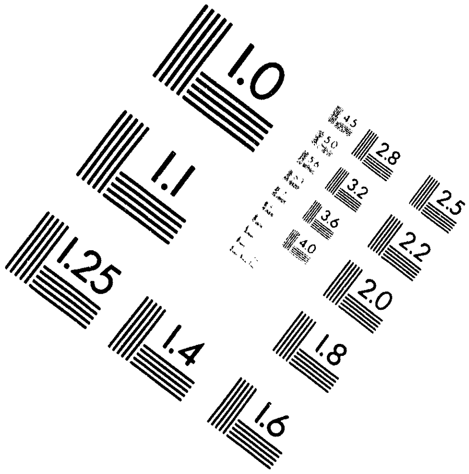


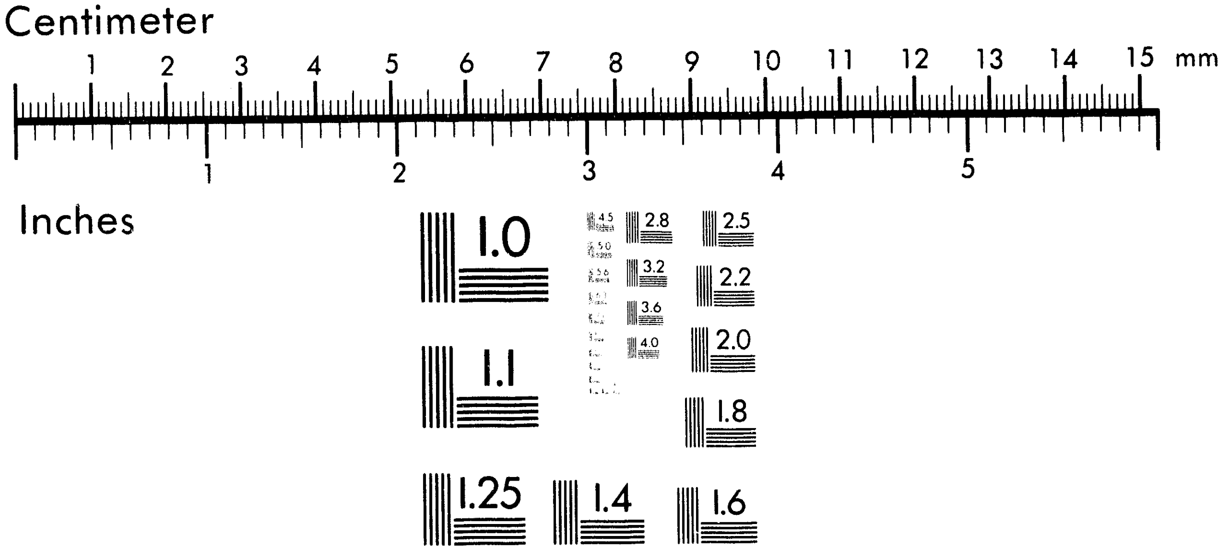
AIIM

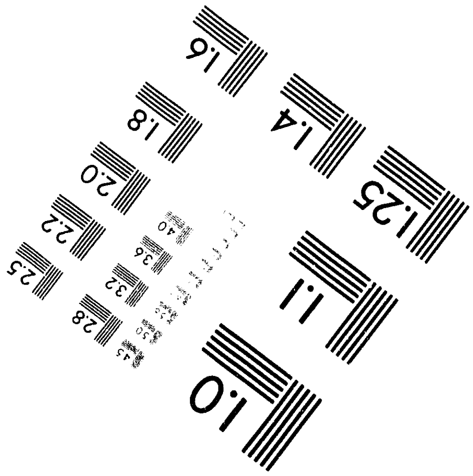
Association for Information and Image Management

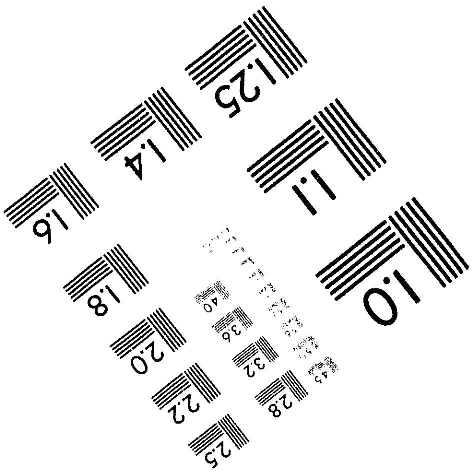
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202









MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 2

CONF-940157--

PNL-SA-24393

PROCEEDINGS OF THE NEURAL NETWORK WORKSHOP FOR THE HANFORD COMMUNITY

P. E. Keller

January 1994

Presented at the
Neural Network Workshop for the Hanford Community
January 26, 1994
Richland, Washingtor

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Contents

Workshop Presentations

Introduction to the Workshop

These proceedings were generated from a series of presentations made at the Neural Network Workshop for the Hanford Community. The abstracts and viewgraphs of each presentation are reproduced in these proceedings. This workshop was sponsored by the Computing and Information Sciences Department in the Molecular Science Research Center (MSRC) at the Pacific Northwest Laboratory (PNL). It was held on Wednesday 26 January 1994 from 8:00 am to 5:00 pm in the Battelle Auditorium. It attracted approximately 90 participants from a variety of organizations including Battelle Pacific Northwest Laboratories, Westinghouse Hanford Company, Boeing Computer Services Richland, Siemans Power Corporation, Systek, Mohr and Associates, NeuroDynamX, Washington State University (Pullman and Tri-Cities campuses), the University of Washington, Eastern Washington University, Eastern Oregon State College, the U.S. Department of Energy, and Bureau of Reclamation.

Artificial neural networks constitute a new information processing technology that is destined within the next few years, to provide the world with a vast array of new products. A major reason for this is that artificial neural networks are able to provide solutions to a wide variety of complex problems in a much simpler fashion than is possible using existing techniques. In recognition of these capabilities, many scientists and engineers are exploring the potential application of this new technology to their fields of study.

An artificial neural network (ANN) can be a software simulation, an electronic circuit, optical system, or even an electro-chemical system designed to emulate some of the brain's rudimentary structure as well as some of the learning processes that are believed to take place in the brain. For a very wide range of applications in science, engineering, and information technology, ANNs offer a complementary and potentially superior approach to that provided by conventional computing and conventional artificial intelligence. This is because, unlike conventional computers, which have to be programmed, ANNs essentially learn from experience and can be trained in a straightforward fashion to carry out tasks ranging from the simple to the highly complex.

Purpose

The purpose of this workshop was to bring together individuals from the Hanford community who potentially have a need to apply artificial neural network technology in their projects. The main objectives of this workshop were:

- to provide an introduction to the field of neural networks
- to explain how neural networks can be used in science and engineering
- to demonstrate how to build a neural network application in software
- to discuss sources of information about neural networks
- to provide a set of example neural network applications of interest to the Hanford community
- to bring together people with interests in neural networks

Workshop Organizing Committee:

Paul E. Keller	(509) 375-2254	pe_keller@pnl.gov
Lars J. Kangas	(509) 375-3905	lj_kangas@pnl.gov
Richard T. Kouzes	(509) 375-6455	rt_kouzes@pnl.gov

Address:
Molecular Science Research Center
Pacific Northwest Laboratory
K1-87
P.O. Box 999
Richland, WA 99352

Facsimile: (509) 375-6631

Workshop Itinerary (Wednesday 26 January 1994)

Tutorial Session

8:00 am - 9:45 am Battelle Auditorium

8:00 am - 8:05 am
Overview of the Workshop, Richard Kouzes, *Pacific Northwest Laboratory, Molecular Science Research Center*

8:05 am - 9:15 am
Introduction to Neural Networks, Paul E. Keller, Pacific *Northwest Laboratory, Molecular Science Research Center* 5

9:15 am - 9:45 am
Developing an Application, Lars J. Kangas, Pacific *Northwest Laboratory, Applied Physics Center* 12

Morning Break

9:45 am - 10:00 am Auditorium Lobby

Overview Session

10:00 am - 12:00 pm Battelle Auditorium
Paul E. Keller, moderator

10:00 am - 10:10 am
Opening Remarks, Richard T. Kouzes, *Pacific Northwest Laboratory, Molecular Science Research Center*

10:10 am - 10:50 am
Developments in Neural Network Chips, Ronald Benson, *NeuroDynamX, Inc.* 15

10:50 am - 11:30 am
Alternatives to Neural Networks: Genetic Algorithms and Non-Linear Biased Regression, Barry M. Wise, *Pacific Northwest Laboratory, Molecular Science Research Center* 20

11:30 am - 12:00 pm
Combining Neural Networks, Sherif Hashem, Pacific *Northwest Laboratory, Molecular Science Research Center* 27

Lunch Break and Poster Session

12:00 pm - 12:40 pm Auditorium Lobby

Afternoon Session #1

12:40 pm - 2:40 pm Battelle Auditorium
Lars J. Kangas, moderator

12:40 pm - 1:10 pm
Parallel and Distributed Gradient Decent Learning, I. Mehr, Zoran Obradovič, R. Venkateswaran, Washington *State University, School of Electrical Engineering and Computer Science* 31

1:10 pm - 1:40 pm
Neural Network Algorithms for VLSI Design and Test Automation, Jack L. Meador, Washington *State University, School of Electrical Engineering and Computer Science* 39

1:40 pm - 2:00 pm
Isotope Identification via Neural Network, Paul Keller, Pacific *Northwest Laboratory, Molecular Science Research Center*........... 45

2:00 pm - 2:20 pm
A Multicomponent Spectrum Decomposition Network, Harry Bell, *U.S. Department of Energy* 49

2:20 pm - 2:40 pm
Neural Network Based Chemical Sensor Systems, Paul Keller, *Pacific Northwest Laboratory, Molecular Science Research Center* 53

Afternoon Break and Poster Session

2:40 pm - 3:00 pm Auditorium Lobby

Afternoon Session #2

3:00 pm - 5:00 pm Battelle Auditorium
Paul E. Keller, moderator

3:00 pm - 3:20 pm
Integrating A Parallel Constructive Neural Network Algorithm with an Expert System, Justin Fletcher, Washington *State University, School of Electrical Engineering and Computer Science* 58

3:20 pm - 3:40 pm
Turbine Engine Diagnosis Artificial Neural Network (TEDANN), Lars J. Kangas, Frank L.

Poster Session

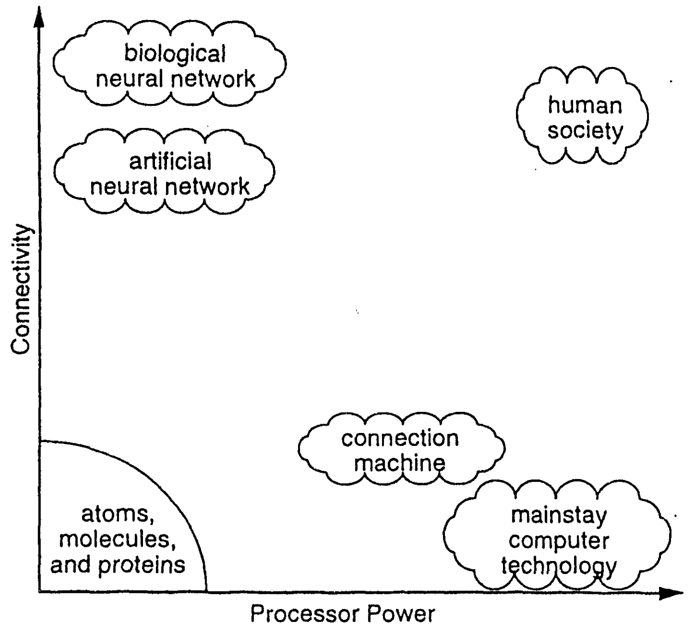
12:00 pm - 12:50 pm
2:50 pm - 3:10 pm **Auditorium Lobby**

Introduction to Neural Networks

Paul E. Keller

Pacific Northwest Laboratory, Molecular Science Research Center,
Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov

Processor Power and Connectivity


biological neural network
artificial neural network
human society
Connectivity
connection machine
atoms, molecules, and proteins
mainstay computer technology
Processor Power

Who is interested in neural networks?

- Biologists and Physiologist
  explore low-level and mid-level brain functions such as memory, sensory systems, and motor functions
- Cognitive Scientists and Psychologists
  model high-level brain functions such as thinking and conscience
- Engineers
  - pattern recognition -design
  - signal processing - modeling
  - process control - robotics
- Computer and Information Scientists
  - explore learning systems
  - apply neural networks to information processing
- Physicists and Chemists
  - model physical and chemical systems
  - pattern recognition
- Doctors and Health Scientists
  - diagnosis - pattern recognition
- Economists, Financial Analysts, and Statisticians
  analyze and predict economic indicators
- Applied Mathematicians
  study neural networks in relation to complex and chaotic systems
- Electrical Engineers and Optical Scientists/Engineers
  develop and build electronic and optical hardware systems to implement artifical neural networks

Where Are Neural Networks Appropriate?

- pattern recognition
- decision
- forecasting
- mapping complex data sets
- modeling when the physical process is not known or understood
- optimization when a "good" solution is acceptable (sub-optimal)
- when the rules of a problem are unknown
- {processing information that humans process well}

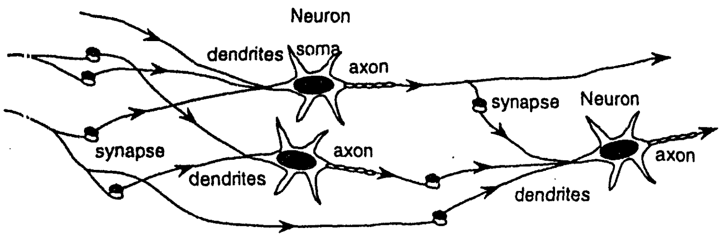
Where Are Neural Networks Not Appropriate?

- when high precision is required
- numerical calculations
- when you already have a good physical model
- when a simpler solution exists

Topics Explored With Neural Networks

Finance (Credit Approval; Predicting Stock; Capital, Currency, and Commodity Markets; Economic Modeling and Analysis; Check Processing)
Business (Forecasting Markets and Sales; Modeling, Sales and Marketing Analysis; Prediction of Workload, Delivery Schedules, Consumer Reaction, etc)
Electrical Engineering (Load Forecasting, Circuit Design, Power System Stability, Optimization of Electric Power Distribution, Adaptive Logic Elements
Communication (Adaptive Signal Processing, Echo Cancellation, Modems, Noise Filtering, Data Compression, Speech and Handwriting Recognition)
Food and Agriculture (Food Inspection, Experiments in Agriculture, Food and Beverage Odor Monitoring, Analysis of Food Quality, etc.)
Optics (Adaptive Focusing, Adaptive Optical Telescopes, Image Classification, Spectral Analysis, Optically Implemented Neural Networks, etc.)
Medicine and Health (Diagnostic Aides -- Myocardial Infarction, Image Analysis -- CT, MRI, Ultrasound, Drug Development, Chemical Analysis)
Education (Teaching Problem Solving, Predicting Student Performance, Teaching Languages, Computer Aided Instruction)
Manufacturing (Industrial Inspection, Process Control, Quality Control, Optimizing Production Schedules, Process Planning and Manufacturing, etc.)
Engineering (Fault Detection, Robotics, Artificial Limb Control; Control, Analysis of Mechanical Systems, Intelligent Alarm Processing, Glass Design)
National Defense (Target Recognition & Tracking, Computer Aided Trans-lation, Pilot Training, Classifying Sonar and Radar, Guidance, Vehicle Control)
Criminal Justice & Law Enforcement (Predicting Parolee Recidivism, Finger Print and Human Face Identification, Detection of Plastic Explosives)
Geology and Geophysics (Oil Exploration, Recognition of Seismic Events, Prediction of Oil Reserves)
Biology (Modeling Biological Systems Including Biological Neural Nets, Cell and Organism Identification, Data Analysis, Optimizing Experimental Results)
Chemistry (Chemical Compound Identification, Spectral Analysis, Chemical Sensor Analysis, Polymer Identification, Determining Structure of Proteins, etc.)
Physics (Modeling Nuclear Systematics, Particle and Radiation Pattern Identification, Gamma Ray Spectroscopy, Modeling in Statistical Mechanics)
Sports (Athletic Training, Picking Horse Race Winners, NFL Score Prediction)
Other Applications (Legal Strategies, Ground Water Quality Control Music Composition andAnalysis, Semantic Feature Analysis, etc.)

Neural Network Fundamentals



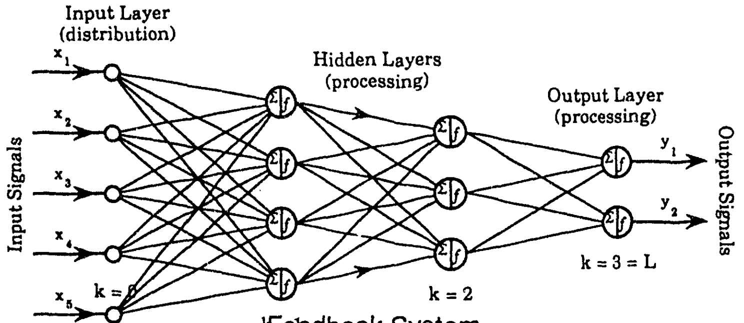
- Simple Model of Biological Neural Mechanisms

neurons	⇔	nonlinear processing elements
synapses	⇔	weighted interconnections between neurons
dendrites/axons	⇔	communication channels
neural network	⇔	system of highly interconnected non-linear processors working in unison

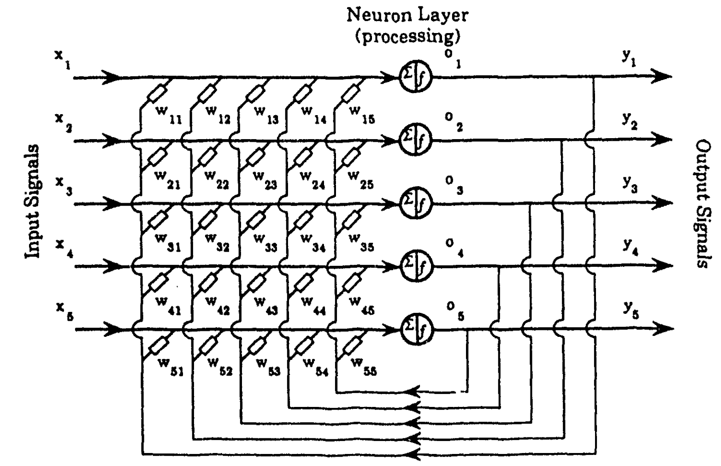
- Artificial Neural Network
  - system constructed from at least the simplistic model of the biological neural network
  - Parallel Distributed Processing (PDP) System
  - Connectionist Model

Neural Network Systems

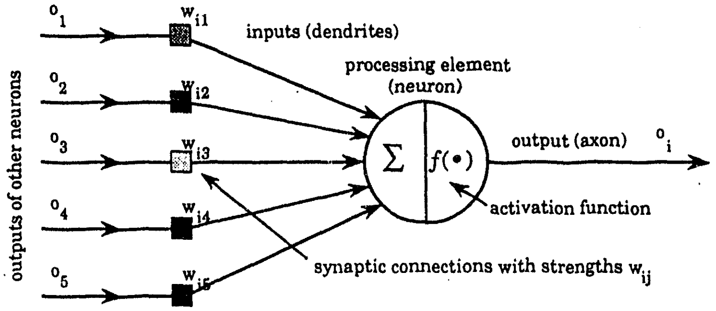
Feedforward System



Feedback System



Individual Neuron and Connected Synapses



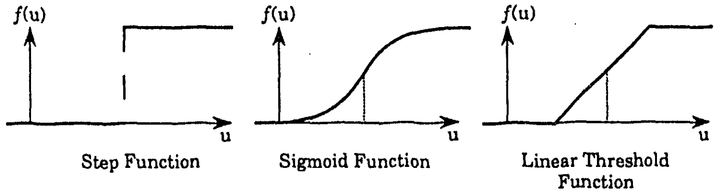
neuron interconnection
- communication
- multiplication

$w_{ij}o_j$

nonlinear processing
- summation
- nonlinear operation, $f(u)$

$f(\sum_{j=1}^{N} w_{ij}o_j)$

Activation Functions



Software Implementation

- Series of Vector-Matrix Multiplications

- Feed Values Into The Network
  $\underline{o}^0 = \underline{x}$ $\{o_j^0 = x_j;$ for all j$\}$

- Propagate To The Next Layer
  $\underline{net}^1 = W^0 o^0$ $\{net_i^1 = \sum w_{ij}^0 o_j^0\}$

- Apply Activation Function
  $\underline{o}^1 = f(\underline{net}^1)$ $\{o_j^1 = f(net_j^1);$ for all j$\}$

- Propagate To The Next Layer
  $\underline{net}^2 = W^1 o^1$ $\{net_i^2 = \sum w_{ij}^1 o_j^1\}$

- Apply Activation Function
  $\underline{o}^2 = f(\underline{net}^2)$ $\{o_j^2 = f(net_j^2);$ for all j$\}$

- Feed Values Out Of The Network
  $\underline{y} = \underline{o}^L$ $\{y_j = o_j^L;$ for all j$\}$

Pattern Recognition

Prediction
- Recognize Trends in Time Series Data

Decision
- Recognize Key Components in a Given Situation

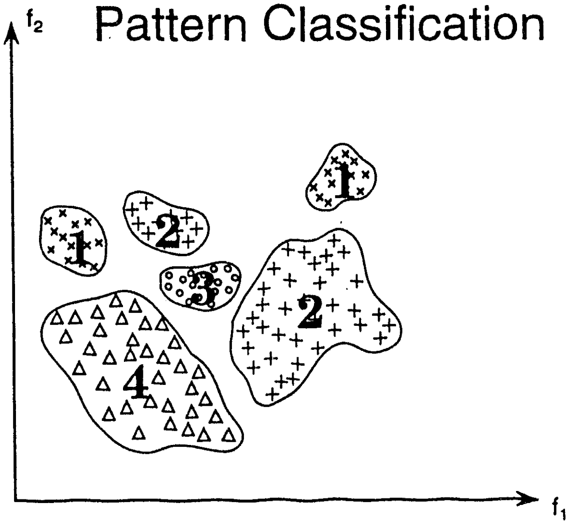
Pattern Classification
- Recognize Objects and Assign Them to the Appropriate Classes

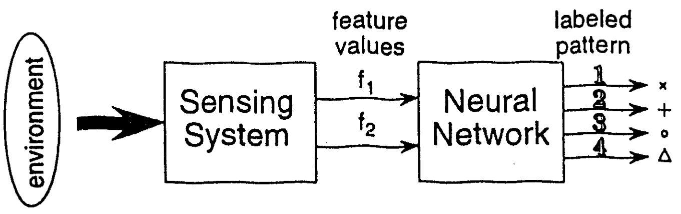
Learning

Supervised
- Example Patterns are Presented to the Neural Network and the Synaptic Weights are Adjusted to Produce the Desired Output
- Maps the Input Pattern to the Desired Output Label
- Analogous to a Student Guided by an Instructor

Unsupervised
- Neural Network Assigns Its Own Labels or Classes
- Useful for Finding Intrinsic Classes or the Underlying Structure of the Data
- Also Known as Self-Organization
- Analogous to a Student Deriving the Lesson Totally on His/Her Own

Pattern Classification

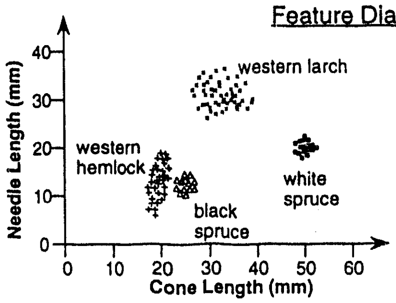




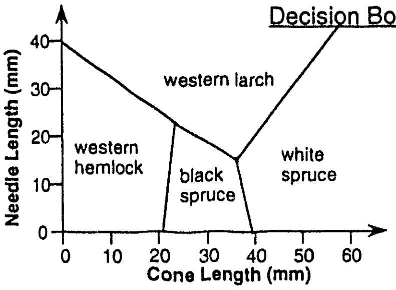
Sensing System
- imaging system, spectrometer, chemical sensor array, etc.

Measurements (Features)
- wavelength, color, voltage, temperature, pressure, intensity, shape, etc.

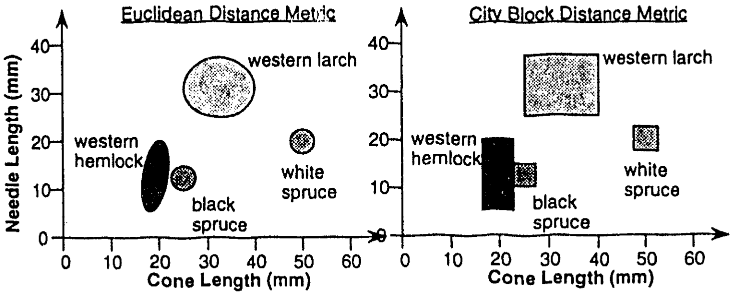
Tree Recognition Example


Feature Diagram

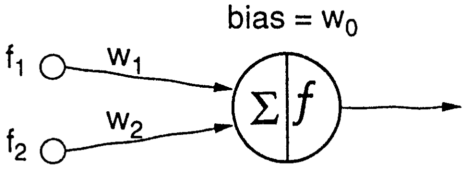
- Measured Values of Sensed Objects
- Two Features Shown Here
  - Needle Length
  - Cone Length


Decision Boundaries

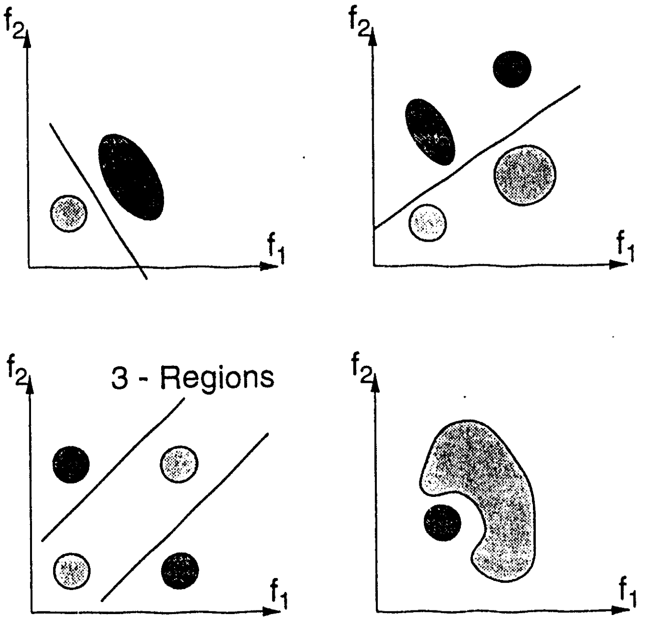
- Boundaries Separate Measurement Values of Different Objects
  - Nearest Neighbor
  - Nearest Mean
  - Euclidean Distance
  - City Block Distance

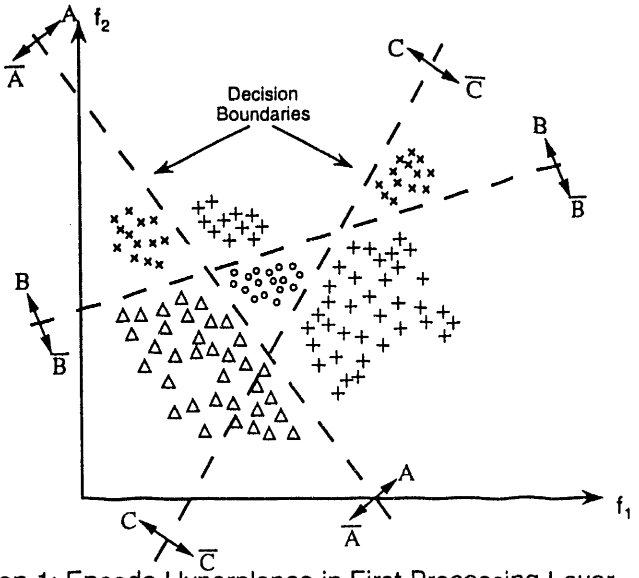

Euclidean Distance Metric

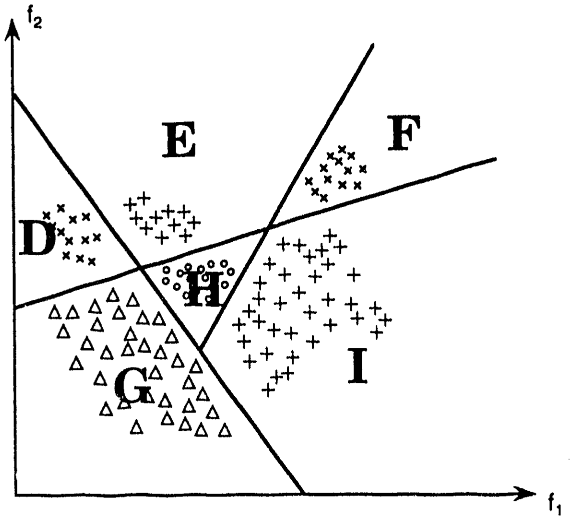
City Block Distance Metric



Net Stimulus = $w_0 + w_1 f_1 + w_2 f_2 = 0$

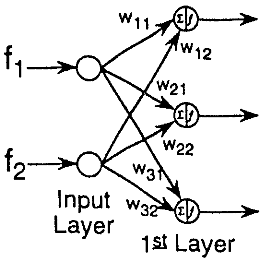






Step 1: Encode Hyperplanes in First Processing Layer

- Hyperplanes encoded by linearly combining inputs
- Each neuron encodes an individual hyperplane in feature space



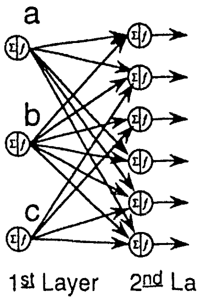
$$a = f(\underset{w_{11}}{1.39}\, f_1 + \underset{w_{12}}{1.00}\, f_2 - \underset{\text{bias}_1}{0.85})$$

$$b = f(\underset{w_{21}}{-0.29}\, f_1 + \underset{w_{22}}{1.00}\, f_2 - \underset{\text{bias}_2}{0.40})$$

$$c = f(-1.81\, f_1 + 1.00\, f_2 + 0.41)$$

Step 2: Encode Convex Regions in Second Processing Layer

- Convex regions encoded by intersecting (and-ing) half-spaces
- Each neuron describes a convex region in feature space



$d = f(-a + b - 0.5)$

$e = f(a + b + c - 2.5)$

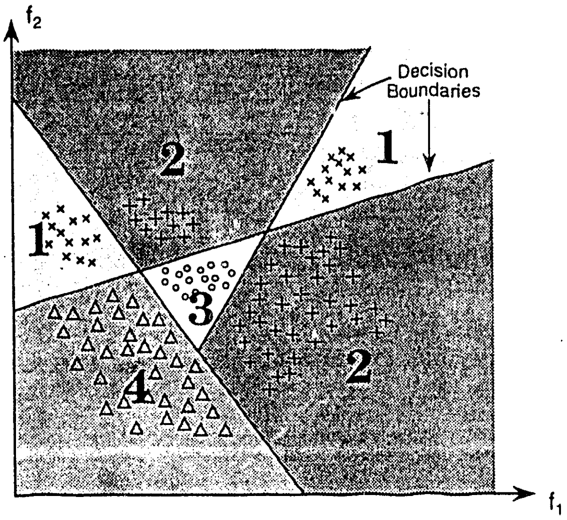
$f = f(b - c - 0.5)$

$g = f(-a - b + 0.5)$

$h = f(a - b + c - 1.5)$

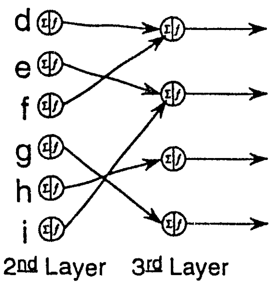
$i = f(a - b - c - 0.5)$

$D = \overline{A} \cap B$
$E = A \cap B \cap C$
$F = B \cap \overline{C}$
$G = \overline{A} \cap \overline{B}$
$H = A \cap \overline{B} \cap C$
$I = A \cap \overline{B} \cap \overline{C}$



Step 3: Combine Regions with the Third Processing Layer

- Arbitrary regions encoded by uniting (or-ing) convex regions
- Each neuron encodes a labeled output



$f(d + f - 0.5)$

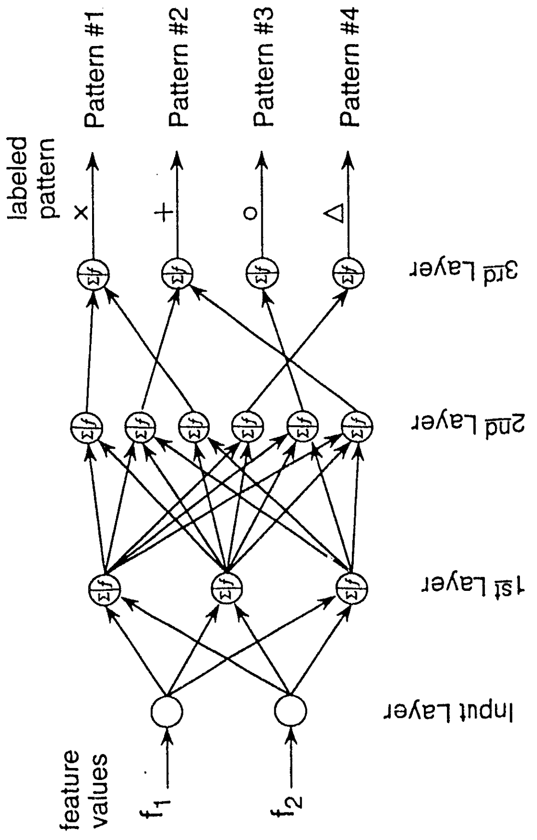
$f(e + i - 0.5)$

$f(h - 0.5)$

$f(g - 0.5)$

$1 = D \cup F$
$2 = E \cup I$
$3 = H$
$4 = G$

Neural Network Pattern Classifier



Artificial Neural Networks

- Hopfield Network

Architecture:	Feedback
Learning:	Supervised (Outer Product Formulation)
Applications:	Associative Memory, Optimization
Input Data:	Binary

- Multilayer Perceptron

Architecture:	Feed Forward
Learning:	Supervised (Backpropagation of Error)
Applications:	Pattern Classification
Input Data:	Continuous

- Boltzmann Machine

Architecture:	Feedback
Learning:	Supervised
Applications:	Optimization
Input Data:	Binary

- Hamming Network

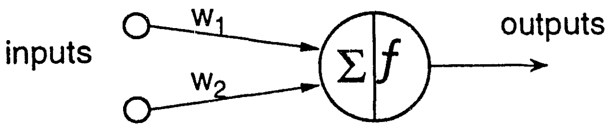
Architecture:	Combination of Feed Forward and Feedback
Learning:	Supervised
Applications:	Pattern Classification
Input Data:	Binary

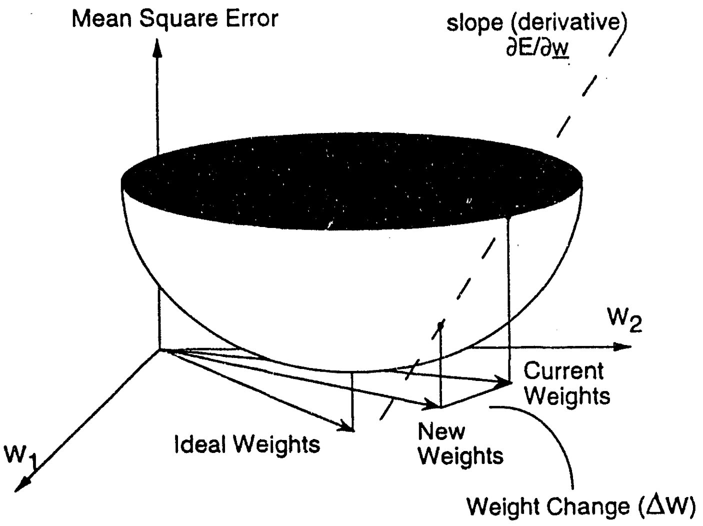
- Kohonen Self-Organizing Feature Map Network

Architecture:	Combination of Feed Forward and Feedback
Learning:	Unsupervised
Applications:	Pattern Clustering
Input Data:	Continuous

- Carpenter-Grossberg Network

Architecture:	Combination of Feed Forward and Feedback
Learning:	Unsupervised (Adaptive Resonance Theory)
Application:	Pattern Clustering
Input Data:	Binary (ART1), Continuous (ART2)





$$\Delta w_{ji} = \eta\, \partial_i\, o_j \quad + \quad \alpha \Delta w_{ji}(n-1)$$

Weight Change

Learning Rate

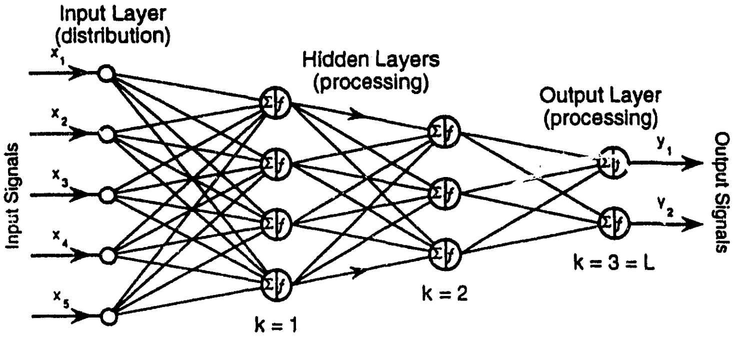
Contains Error

Output

Momentum

Previous Update

Backpropagation Algorithm



Batch Learning Process

Pick a labeled pattern from the training set and present it to the network. (training pattern ⇒ $\underline{x}^p$, target output ⇒ $\underline{t}^p$)

Propagate data forward and generate the output classification. (output ⇒ $\underline{y}^p$)

Calculate error between target classification and actual classification.

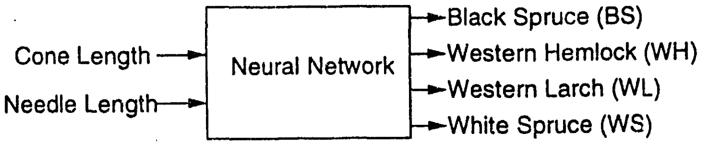
$$\text{Error} = \frac{1}{2}\sum_{j=1}^{N}(t_j^p - y_j^p)^2$$

Propagate error backwards through network and calculate changes to the synaptic weights that will reduce output error using the generalized delta rule.

If there are more patterns in the training set, loop back.

Update synaptic weight values in the network.

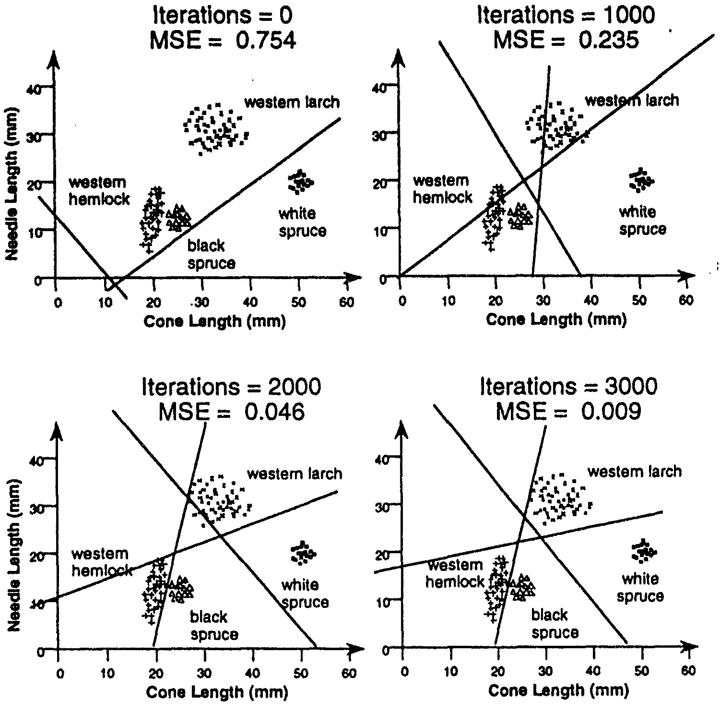
If output error high or maximum number of iterations not met, then loop back



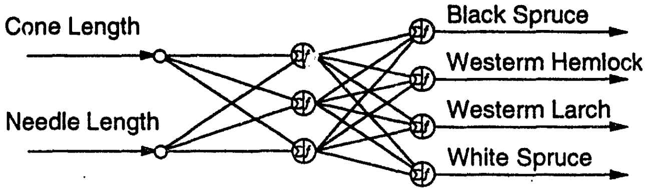
Training Set

Cone Length	Needle Length	Tree	BS	WH	WL	WS
25 mm	11 mm	Black Spruce	1	0	0	0
26 mm	11 mm	Black Spruce	1	0	0	0
26 mm	10 mm	Black Spruce	1	0	0	0
24 mm	9 mm	Black Spruce	1	0	0	0
20 mm	13 mm	Western Hemlock	0	1	0	0
21 mm	14 mm	Western Hemlock	0	1	0	0
19 mm	8 mm	Western Hemlock	0	1	0	0
21 mm	20 mm	Western Hemlock	0	1	0	0
28 mm	30 mm	Western Larch	0	0	1	0
37 mm	31 mm	Western Larch	0	0	1	0
33 mm	33 mm	Western Larch	0	0	1	0
32 mm	28 mm	Western Larch	0	0	1	0
51 mm	19 mm	White Spruce	0	0	0	1
50 mm	20 mm	White Spruce	0	0	0	1
52 mm	20 mm	White Spruce	0	0	0	1
51 mm	21 mm	White Spruce	0	0	0	1

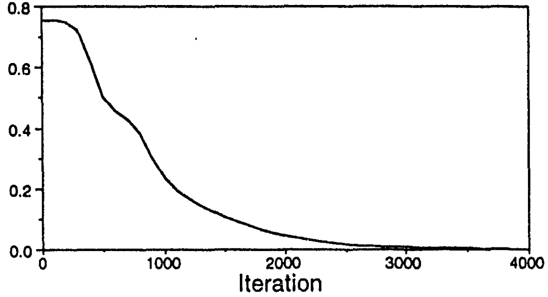
Neural Network Approach to Tree Recognition

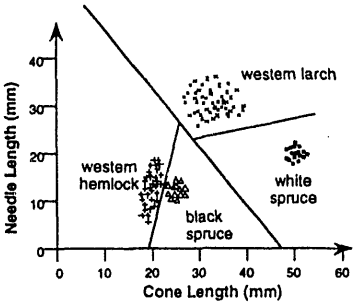


Results with Neural Network



Mean Square Error (MSE)

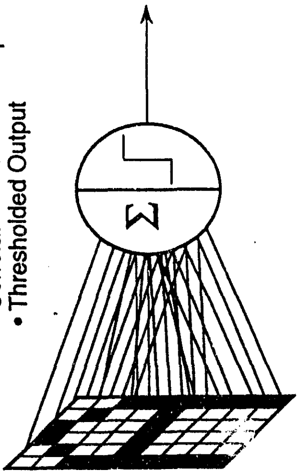


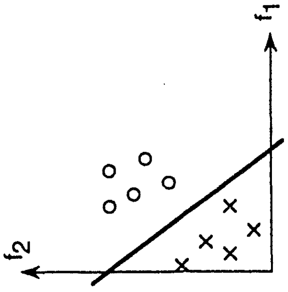


Perceptron

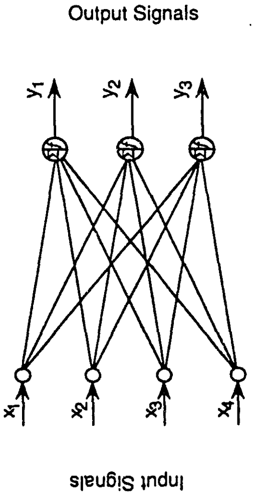
Applications: Typed-character recognition
Developers: Frank Rosenblatt, Cornell University, 1957
Architecture: Single Layer Feed-forward
Activation: Step for Perceptron Learning Rule
Linear or Linear Threshold for a-LMS
Learning Rule: Perceptron Learning Rule
Widrow-Hoff or Least Mean Squares (a-LMS)
Method: Supervised
Input Data: Continuous
Strengths: Simple System
Linear classification
Weaknesses: Cannot Recognize Complex Characters
Sensitive to Scale, Translation, Rotation

- Correlation with an exemplar
- Thresholded Output



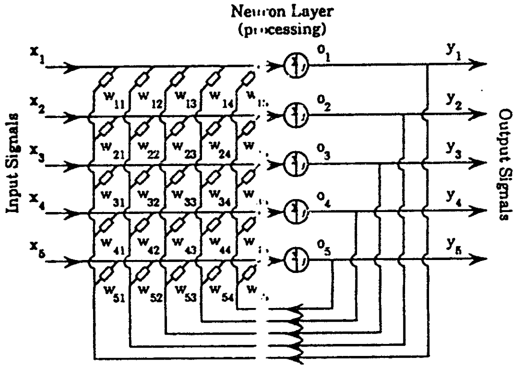


Optimal Linear Associative Memory (OLAM)



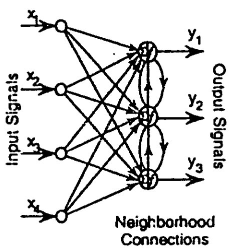
Applications: Associative memory
Pattern classification
Composition analysis
Developers: Teuvo Kohonen, University of Helsinki, 1972
Architecture: Single Layer Feed-Forward
Activation: Linear
Learning Rule: Matrix Orthogonalization
Method: Deterministic
Input Data: Continuous
Strengths: Easily trained (simple algorithm)
Good at composition analysis
Weaknesses: Limitation on the number of patterns stored
Linear classification

Hopfield Networks

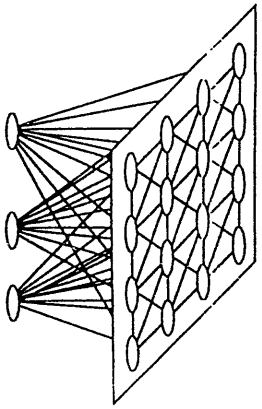


Applications:	<u>Associative Memory</u>	<u>Optimization</u>
Architecture:	Feedback	Feedback
Developers:	John J. Hopfield, 1982	John J. Hopfield and David Tank, 1985
Learning:	Outer-Product Formulation	Energy Function Encoded in Weights
Method:	Deterministic	Deterministic
Input Data:	Binary or Continous	Binary Coded
Strengths:	Can Retrieve Incomplete or Noisy Patterns	Sometimes Finds Good Solutions to NP-Complete Problems
Weaknesses:	Low Memory Capacity.	Easily Trapped in Local Minima Tends to Oscillate

Kohonen Self-Organizing Feature Map

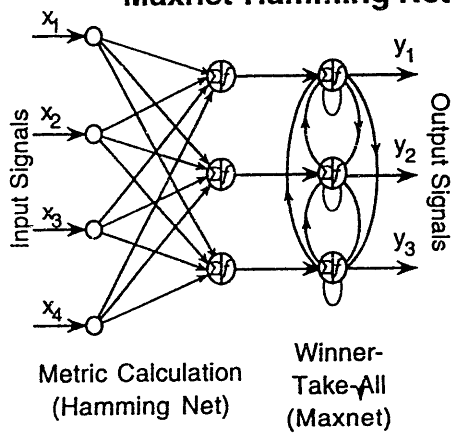


Applications:	Pattern Clustering
Architecture:	Single Processing Layer Interconnected to Input with Neighborhood Connections
Learning:	Unsupervised (Self-Organizing)
Method:	Euclidean Distance Metric
Input Data	Continuous
Strengths:	Can Detect Inconsistencies in Data, Good for Classification
Weaknesses:	Lack of Theory for Developing Cluster Size, Cluster Instability



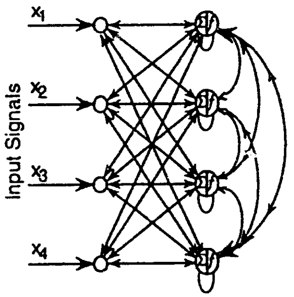
Topological Mapping

Maxnet-Hamming Net



Applications:	Pattern classification
Developer:	R.P. Lipmann, MIT Lincoln Labs, 1987
Architecture:	Feed-Forward Followed by a Lateral Inhibition Layer
Learning:	Hamming Distance Metric Formulation
Method:	Deterministic
Input Data:	Binary
Strengths:	Implements optimum classifier for binary patterns High storage capacity
Weakness:	Binary data

Adaptive Resonance Theory (ART)



Applications:	Pattern clustering (complicated patterns)
Developers:	Stephen Grossberg, Boston University, 1978-86 Gail Carpenter, Northeastern University
Architecture:	Bidirectional between input and output layers and output has lateral inhibition
Learning Rule:	Adaptive Resonance Theory (ART)
Method:	Unsupervised
Input Data:	Binary (ART1), Continuous (ART2, ART2a, ART3)
Strengths:	Able to learn new patterns (form new categories) Very sophisticated Most biologically plausible of the simple models
Weaknesses:	Sensitive to translation, distortion, and scale Exemplars can change over time Likes to see pattern for long time

Developing an Application

Lars J. Kangas

Pacific Northwest Laboratory, Applied Physics Center, Computer Science Department
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-3905 fax: (509) 375-6631 internet: lj_kangas@pnl.gov

1. Collect data with instances of every class
2. Convert data to a form acceptable to the ANN simulator
3. Split data into a training, validation and testing sets
4. Train network on training set until error in the validation set is minimized
5. Test network with testing set to establish the performance

itle: Glass Identification Database

ocation: anonymous ftp to ics.uci.edu /pub/machine-learning-databases/glass

ources:
(a) Creator: B. German
– Central Research Establishment
Home Office Forensic Science Service
Aldermaston, Reading, Berkshire RG7 4PN
(b) Donor: Vina Spiehler, Ph.D., DABFT
Diagnostic Products Corporation
(213) 776-0180 (ext 3014)

urpose:
The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified!

tribute Information:
1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)
-- 1 building_windows_float_processed
-- 2 building_windows_non_float_processed
-- 3 vehicle_windows_float_processed
-- 4 vehicle_windows_non_float_processed (none in this database)
-- 5 containers
-- 6 tableware
-- 7 headlamps

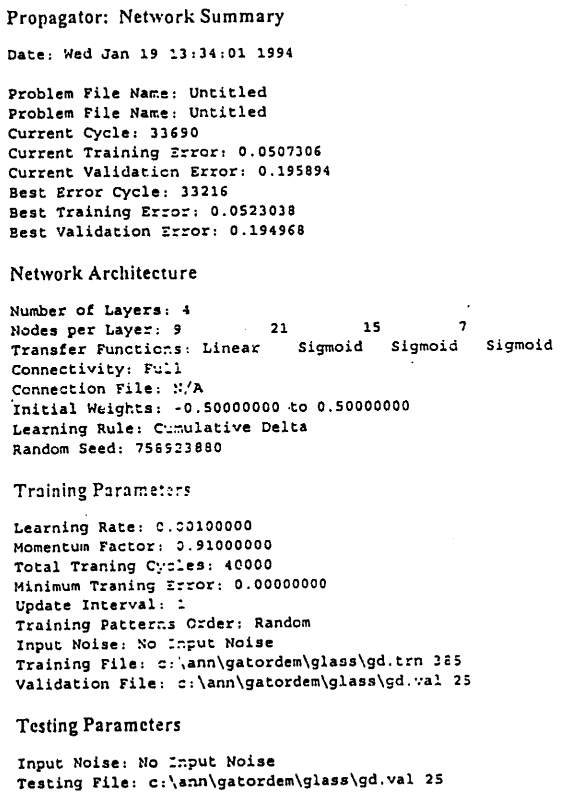
Summary Statistics:

Attribute:	Min	Max	Mean	SD	Correlation with class
2. RI:	1.5112	1.5339	1.5184	0.0030	-0.1642
3. Na:	10.73	17.38	13.4079	0.8166	0.5030
4. Mg:	0	4.49	2.6845	1.4424	-0.7447
5. Al:	0.29	3.5	1.4449	0.4993	0.5988
6. Si:	69.81	75.41	72.6509	0.7745	0.1515
7. K:	0	6.21	0.4971	0.6522	-0.0100
8. Ca:	5.43	16.19	8.9570	1.4232	0.0007
9. Ba:	0	3.15	0.1750	0.4972	0.5751
10. Fe:	0	0.51	0.0570	0.0974	-0.1879

Class Distribution: (out of 214 total instances)
- -- 163 Window glass (building windows and vehicle windows)
  - -- 87 float processed
    - -- 70 building windows
    - -- 17 vehicle windows
  - -- 76 non-float processed
    - -- 76 building windows
    - -- 0 vehicle windows
- -- 51 Non-window glass
  - -- 13 containers
  - -- 9 tableware
  - -- 29 headlamps

```
63 1.52172 13.51 3.86 0.88 71.79 0.23 9.54 0.00 0.11 1
64 1.52227 14.17 3.81 0.78 71.35 0.00 9.69 0.00 0.00 1
65 1.52172 13.48 3.74 0.90 72.01 0.18 9.61 0.00 0.07 1
66 1.52099 13.69 3.59 1.12 71.96 0.09 9.40 0.00 0.00 1
67 1.52152 13.05 3.65 0.87 72.22 0.19 9.85 0.00 0.17 1
68 1.52152 13.05 3.65 0.87 72.32 0.19 9.85 0.00 0.17 1
69 1.52152 13.12 3.58 0.90 72.20 0.23 9.82 0.00 0.16 1
70 1.52300 13.31 3.58 0.82 71.99 0.12 10.17 0.00 0.03 1
71 1.51574 14.86 3.67 1.74 71.87 0.16 7.36 0.00 0.12 2
72 1.51848 13.64 3.87 1.27 71.96 0.54 8.32 0.00 0.32 2
73 1.51593 13.09 3.59 1.52 73.10 0.67 7.83 0.00 0.00 2
74 1.51631 13.34 3.57 1.57 72.87 0.61 7.89 0.00 0.00 2
75 1.51596 13.02 3.56 1.54 73.11 0.72 7.90 0.00 0.00 2
76 1.51590 13.02 3.58 1.51 73.12 0.69 7.96 0.00 0.00 2
77 1.51645 13.44 3.61 1.54 72.39 0.66 8.03 0.00 0.00 2
78 1.51627 13.00 3.58 1.54 72.83 0.61 8.04 0.00 0.00 2
79 1.51613 13.92 3.52 1.25 72.88 0.37 7.94 0.00 0.14 2
80 1.51590 12.82 3.52 1.90 72.86 0.69 7.97 0.00 0.00 2
81 1.51592 12.86 3.52 2.12 72.66 0.69 7.97 0.00 0.00 2
82 1.51593 13.25 3.45 1.43 73.17 0.61 7.86 0.00 0.00 2
```

```
0.3704 0.3714 0.7416 0.3084 0.4554 0.0966 0.3448 0.0000 0.2157
0.9000 0.1000 0.1000 0.1000 0.1000 0.1000 0.1000
0.2652 0.3308 0.7884 0.4143 0.5625 0.1031 0.2416 0.0000 0.4118
0.1000 0.9000 0.1000 0.1000 0.1000 0.1000 0.1000
0.2360 0.3143 0.7840 0.5016 0.5446 0.1111 0.2361 0.0000 0.0000
0.1000 0.9000 0.1000 0.1000 0.1000 0.1000 0.1000
0.3316 0.3774 0.8686 0.3489 0.4500 0.0886 0.2677 0.0000 0.1961
0.1000 0.9000 0.1000 0.1000 0.1000 0.1000 0.1000
0.3104 0.4211 0.7595 0.3832 0.3982 0.0934 0.3123 0.0000 0.0000
0.1000 0.1000 0.9000 0.1000 0.1000 0.1000 0.1000
0.2920 0.2436 0.6058 0.4174 0.5464 0.1127 0.3532 0.0000 0.0000
0.1000 0.9000 0.1000 0.1000 0.1000 0.1000 0.1000
0.3012 0.2767 0.7728 0.3396 0.6393 0.0966 0.2900 0.0000 0.1176
0.9000 0.1000 0.1000 0.1000 0.1000 0.1000 0.1000
0.3876 0.2872 0.0000 0.4237 0.7036 0.0612 0.5669 0.0000 0.0000
0.1000 0.1000 0.1000 0.1000 0.9000 0.1000 0.1000
0.3408 0.5053 0.4878 0.4268 0.5107 0.0000 0.3615 0.0000 0.0000
0.1000 0.1000 0.1000 0.1000 0.1000 0.9000 0.1000
0.2844 0.5253 0.0000 0.5576 0.6339 0.0000 0.2965 0.5302 0.0000
0.1000 0.1000 0.1000 0.1000 0.1000 0.1000 0.9000
0.2696 0.3098 0.7840 0.3894 0.6339 0.1063 0.2296 0.0000 0.0000
0.1000 0.9000 0.1000 0.1000 0.1000 0.1000 0.1000
```



Propagator: Network Summary

```
Date: Wed Jan 19 13:34:01 1994

Problem File Name: Untitled
Problem File Name: Untitled
Current Cycle: 33690
Current Training Error: 0.0507306
Current Validation Error: 0.195894
Best Error Cycle: 33216
Best Training Error: 0.0523038
Best Validation Error: 0.194968
```

Network Architecture

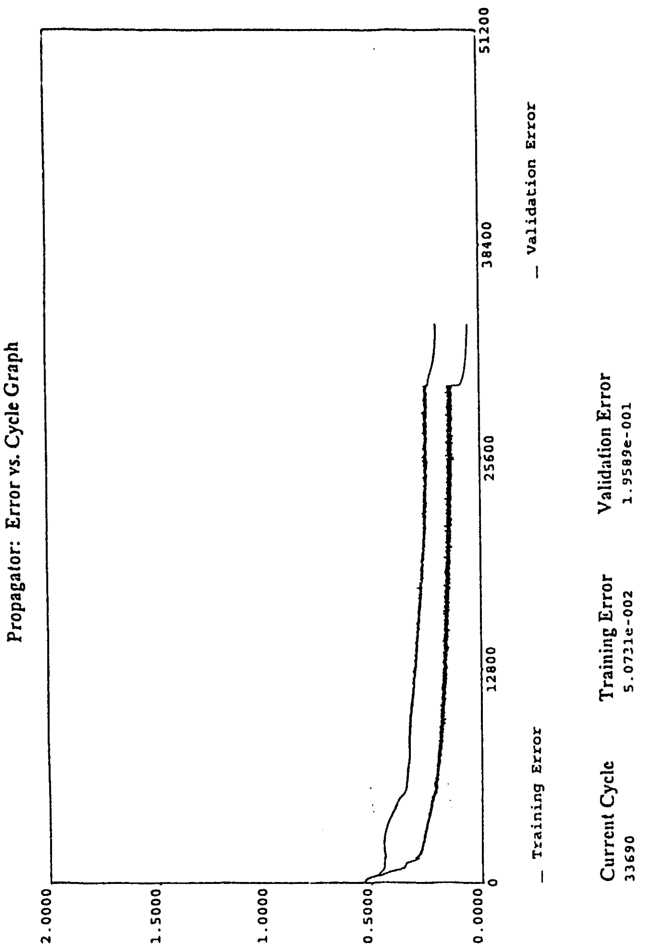
```
Number of Layers: 4
Nodes per Layer: 9          21          15          7
Transfer Functions: Linear    Sigmoid    Sigmoid    Sigmoid
Connectivity: Full
Connection File: N/A
Initial Weights: -0.50000000 to 0.50000000
Learning Rule: Cumulative Delta
Random Seed: 756923880
```

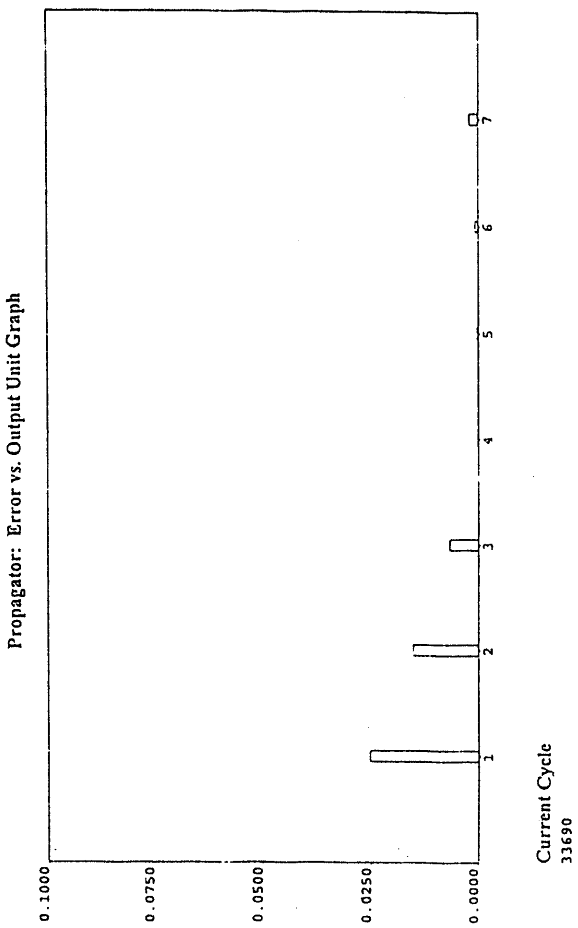
Training Parameters

```
Learning Rate: 0.00100000
Momentum Factor: 0.91000000
Total Traning Cycles: 40000
Minimum Traning Error: 0.00000000
Update Interval: 1
Training Patterns Order: Random
Input Noise: No Input Noise
Training File: c:\ann\gatordem\glass\gd.trn 385
Validation File: c:\ann\gatordem\glass\gd.val 25
```

Testing Parameters

```
Input Noise: No Input Noise
Testing File: c:\ann\gatordem\glass\gd.val 25
```





CONFUSION MATRIX

(Percent correct)

		Neural Network Classification						
		Class1	Class2	Class3	Class4	Class5	Class6	Class7
	Class1	87.5	12.5	0.0	0.0	0.0	0.0	0.0
	Class2	10.0	80.0	0.0.	0.0	0.0	10.0	0.0
Actual Classification	Class3	33.3	33.3	33.3	0.0	0.0	0.0	0.0
	Class4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Class5	0.0	0.0	0.0	0.0	100.0	0.0	0.0
	Class6	0.0	0.0	0.0	0.0	0.0	100.0	0.0
	Class7	0.0	0.0	0.0	0.0	0.0	0.0	100.0

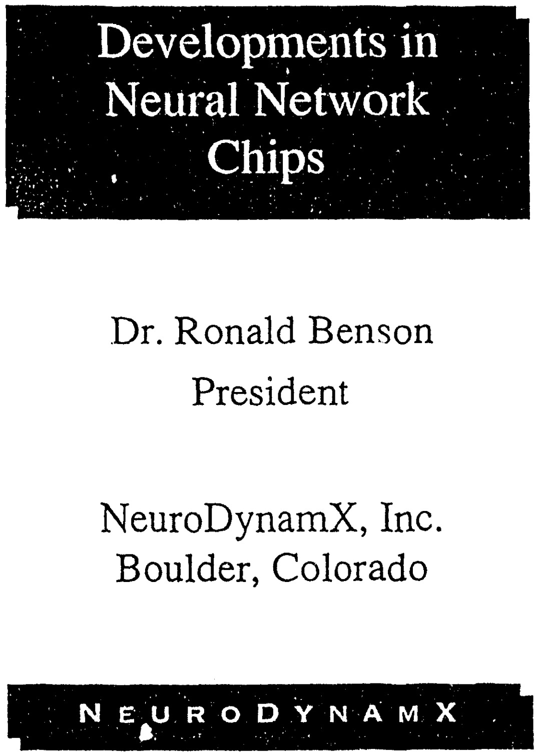
Chance = 100 / 7 = 14.3%

Developments in Neural Network Chips

Dr. Ronald Benson, President

NeuroDynamX, Inc.
P.O. Box 323, Boulder, CO 80306
phone: (303) 442-3539 fax: (303) 442-2854 internet: ron@ndx.com

Hardware implementations of artificial neural networks can yield substantial performance gains over software simulations developed and deployed on conventional von Neumann computers. This is due to the inherently parallel processing architectures of artificial neural networks. Dr. Benson will survey recent developments in VLSI integrated circuits that exploit both this parallelism and the low power requirements of sub-threshold CMOS. Applications include motion-sensitive artificial retinas and artificial cochleas for auditory signal processing. Dr. Benson will also discuss neural implementations for data fusion and signal blending.

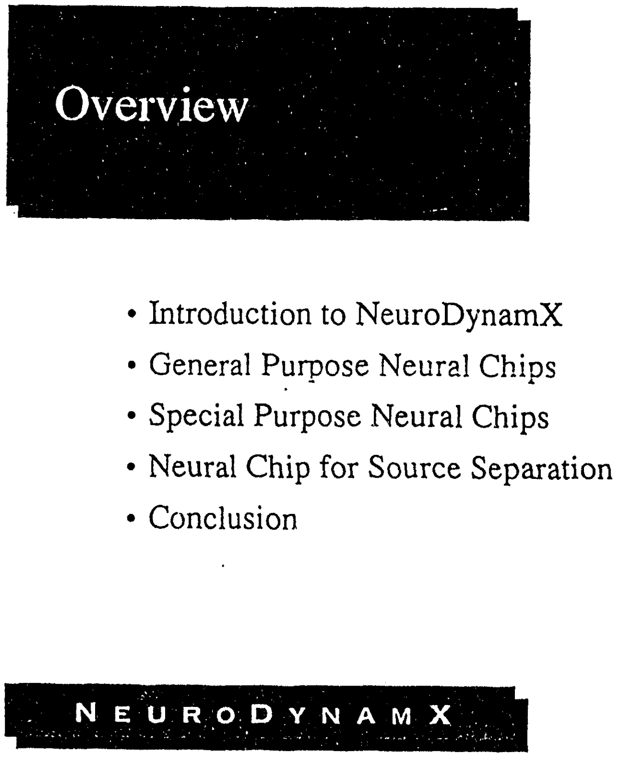


Developments in Neural Network Chips

Dr. Ronald Benson
President

NeuroDynamX, Inc.
Boulder, Colorado

NEURODYNAMX

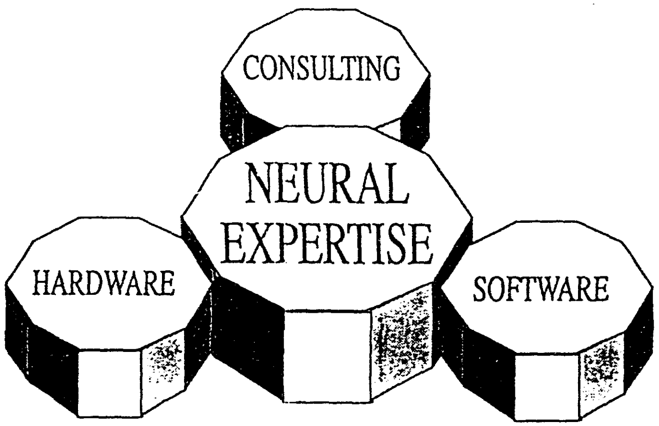


Overview

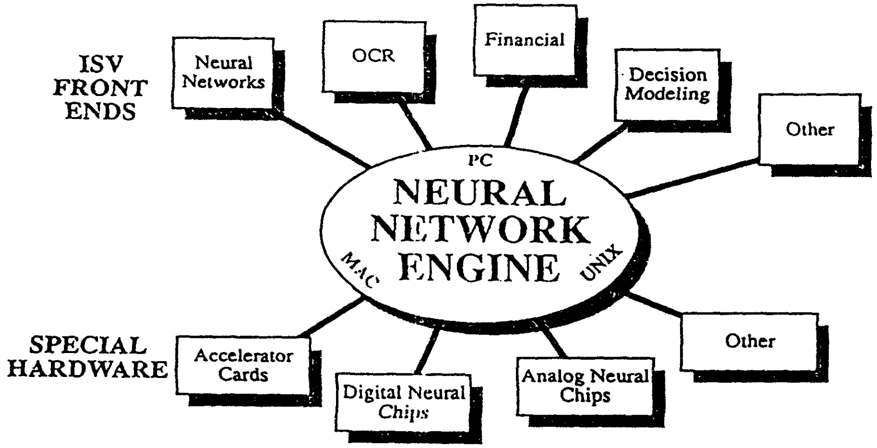
- Introduction to NeuroDynamX
- General Purpose Neural Chips
- Special Purpose Neural Chips
- Neural Chip for Source Separation
- Conclusion

NEURODYNAMX

Corporate Structure


CONSULTING
NEURAL EXPERTISE
HARDWARE
SOFTWARE

NDX Software Concept


ISV FRONT ENDS
Neural Networks
OCR
Financial
Decision Modeling
Other
PC
NEURAL NETWORK ENGINE
MAC
UNIX
SPECIAL HARDWARE
Accelerator Cards
Digital Neural Chips
Analog Neural Chips
Other

NEURODYNAMX

NDX Hardware Concept

- Sensory Processing
- Computationally Efficient
- Input Fusion
- Robust

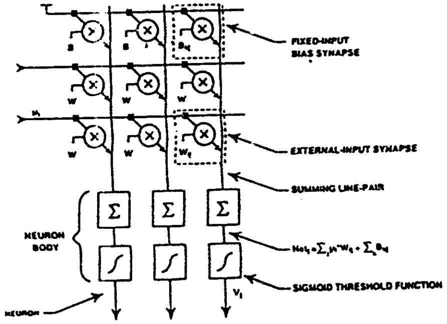
General Purpose Neural Chips

- Connections Per Second
- Connection Updates Per Second
- Number of Connections
- Architecture: Fully Connected
- Power Hungry
- ETANN, CNAPS, Neural Accelerators Pineda Chip, Ni1000

NEURODYNAMX

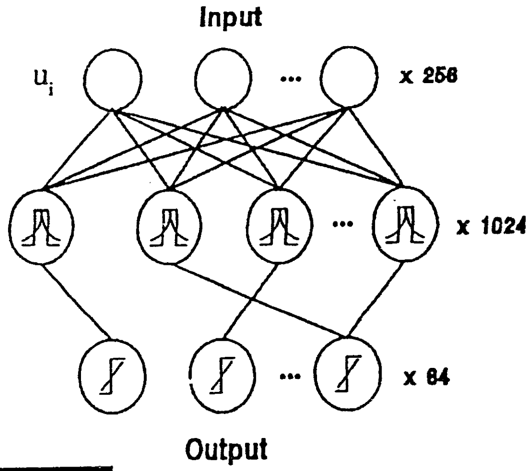
ETANN: Intel, NeuroDynamX

- Two Billion CPS
- Slow Weight Updates
- 10,000 Weights
- 64 Neurons, 128 Inputs



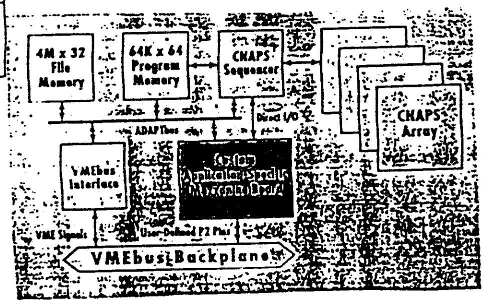
Ni1000: Intel, Nestor, NeuroDynamX

Clock Frequency	33MHz, 25MHz
Performance	
DCU	20GOPS
FPU	160MFbps
uC	20MIPS
Array Access BW	10Gwords/sec (5bil word)
Prototype Vector Size	256 Dim x 5bil
Number of Prototypes	1024 (256 Dim) - 8192 (32 Dir



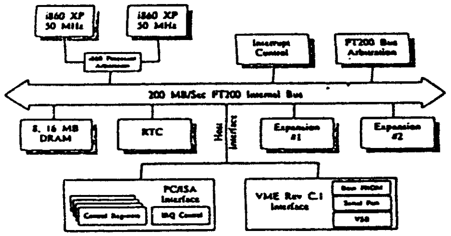
CNAPS: Adaptive Solutions

Characteristics	CNAPS/VME-64	CNAPS/VME-128	CNAPS/VME-256
Number of PNs	64	128	256
Total PN Memory (Each PN has 4096 bytes)			
1-Bit	2,097,152	4,194,304	8,388,608
8-Bit	262,144	524,283	1,048,576
16-Bit	131,072	262,144	524,288
Performance			
Multiply/Accumulates	1.28 Billion	2.56 Billion	5.12 Billion
Backpropagation Feedforward (CPS)	1.16 Billion	2.27 Billion	4.30 Billion
Backpropagation Learning (CUPS)	293 Million	573 Million	1.06 Billion

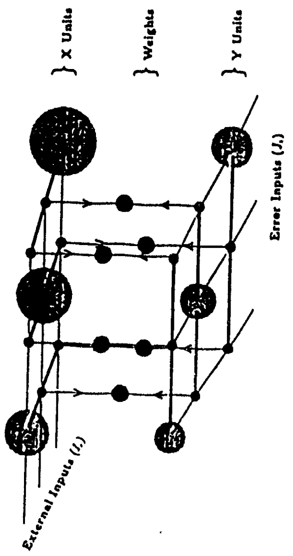


Neural Accelerators: NeuroDynamX

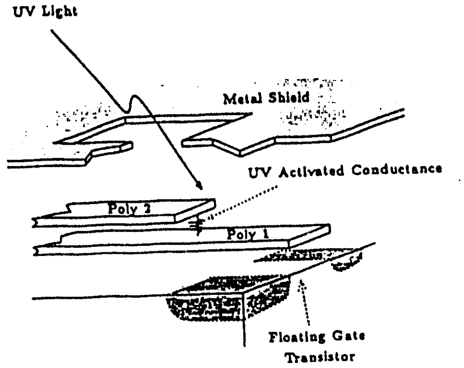
- i860 Accelerator
- 45 Million CPS
- 15 Million CUPS
- Network Size Limited Only by Memory

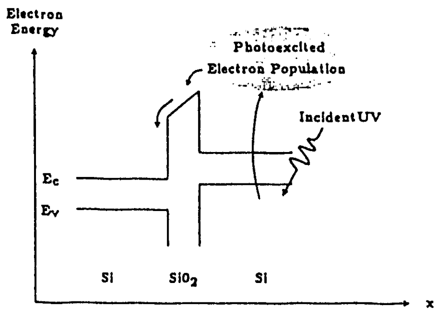


Pineda Chip: Caltech

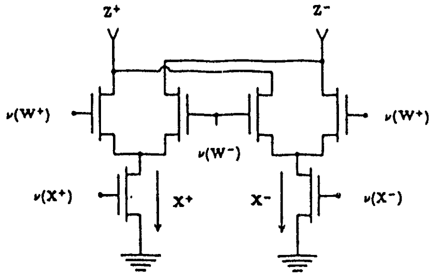


NEURODYNAMX





THE SYNAPSE



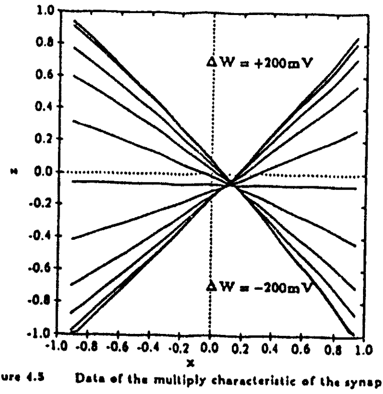
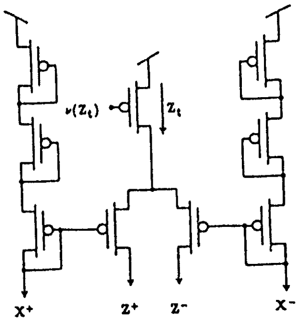
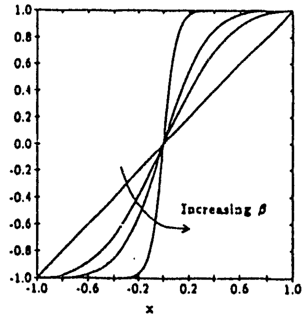


Figure 4.5 Data of the multiply characteristic of the synapse.

THE SIGMOID

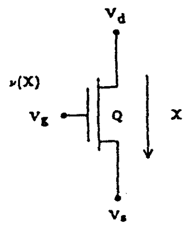




Special Purpose Neural Chips

- Biologically Inspired
- Low Power - Subthreshold CMOS
- Silicon Retinas
- Silicon Cochleas

Low Power - Subthreshold CMOS



Neural Chip for Source Separation

- Herault-Jutten Algorithm

$$
\begin{aligned}
E(t) &= AX(t) \\
S(t) &= E(t) - CS(t) = [I + C]^{-1} AX(t) \\
\frac{dc_{ij}(t)}{dt} &= \epsilon f(S_i(t)) g(S_j(t))
\end{aligned}
$$

where $E(t)$ is the measured signal
A is the mixing matrix
$X(t)$ is the actual signal
and $S(t)$ is the separated signal.

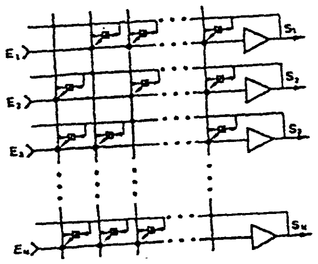
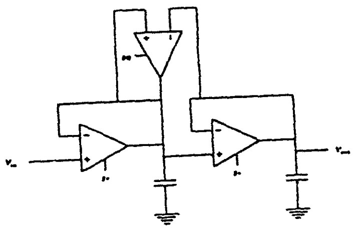


Figure 1: The N x N network architecture.

NEURODYNAMX

Silicon Cochleas



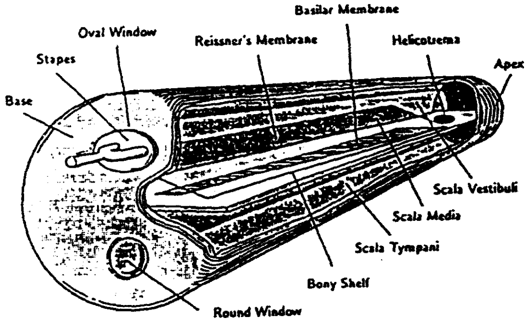


Figure 2.3 The unrolled cochlea, simplified to emphasize the bony shelf and widening of the basilar membrane. Adapted from Cole and Chadwick

NEURODYNAMX

Alternatives to Neural Networks: Genetic Algorithms and Non Linear Biased Regression

Barry M. Wise

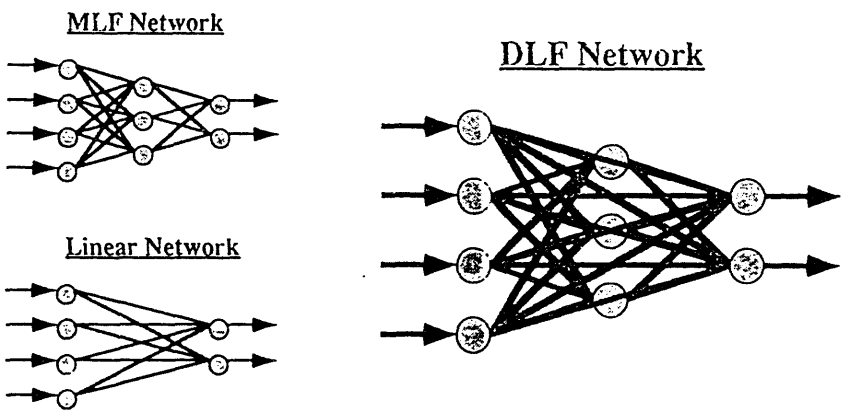
Battelle Pacific Northwest Laboratories
Molecular Science Research Center

Neural Network Workshop
for the Hanford Community
January 26, 1994

Overview

- Modelling techniques
  - DLF Artificial Neural Networks
  - Genetic Algorithm
  - Partial Least Squares
    - linear
    - polynomial inner relation
    - spline inner relation
  - Locally Weighted Regression
- Test system
- Model forms
- Results
- Conclusions

Neural Networks



DLF-ANN Training Algorithm

- Use Sequential Quadratic Programming
  - faster
  - handles multiple functionality well
  - possible to incorporate constraints
- Large nets use Conjugate Gradient methods
- Global minimization procedures?

Genetic Algorithm

- Basic genetic algorithm for optimization:
  - encode potential solutions as binary "genes"
  - initialize with random population
  - test population against "fitness" criteria
  - let fit solutions live and breed, let unfit die
  - stop when population or solutions converge
- In our example
  - use to determine non-linear model structure
  - many possible model forms
  - fitness criteria is predictive ability of model, not fit to data

GA for Model Structure Selection

- Example: suppose z = F(x,y)
- List all possible model terms
  - x y x^2 y^2 x^3 y^3 xy
- Produce random initial population of models, 1 indicates term is used, 0 if term not used:
  - 1 1 0 1 0 0 1
  - 0 1 0 1 0 1 1
  - 1 1 1 1 1 0 0 etc.
- Fit models to part of data set by least squares
- Test models' ability to predict remaining data
- Calculate prediction error, rank models

"Breeding" Models

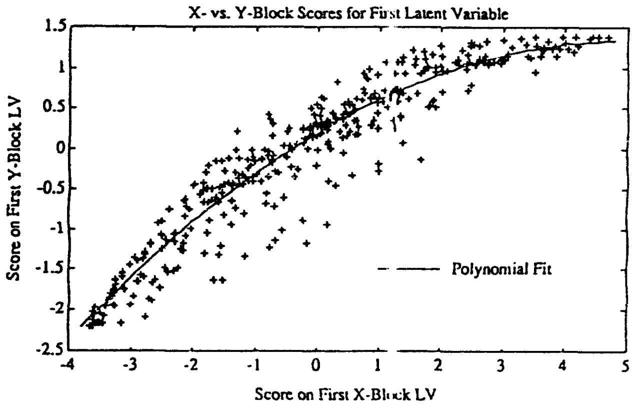
- Randomly select pairs of "fittest" models
- Randomly select "crossover point" and twist
  1 1 | 0 1 0 0 1 --> 1 1 0 1 0 1 1
  0 1 | 0 1 0 1 1 --> 0 1 0 1 0 0 1
- New population consists of fittest models and their offspring
- Check for convergence of population
- Allow for random "mutation" of genes

Partial Least Squares (PLS)

- PLS regression decomposes the matrix of independent variables (U) into linear combinations of variables with greatest covariance with the dependent variables
- PLS algorithm captures the most remaining covariance between U-block scores and y at each step
- PLS factors or Latent Variables are calculated step by step
- Cross-validation is used to determine optimum number of LVs to retain for best prediction

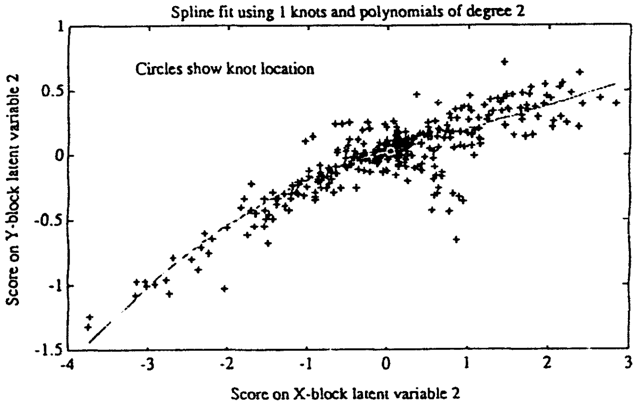
The PLS Inner-Relation

- PLS usually linear--a single coefficient relates "scores" from independent and dependent block
- Inner-relation can be made non-linear, such as a polynomial used here -- PolyPLS


X- vs. Y-Block Scores for First Latent Variable

Spline-Partial Least Squares (SPL_PLS)

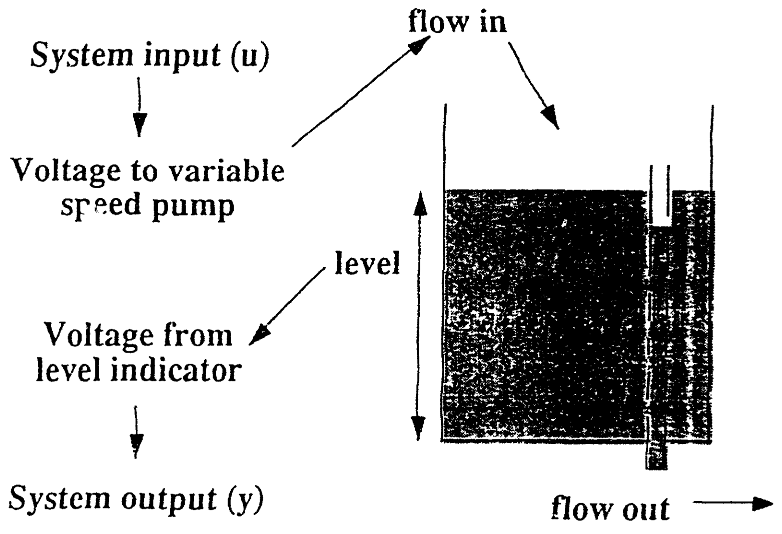
- Inner-relation can also be spline of abritrary degree and number of knots
- SPL_PLS algorithm used here iterates to obtain maximum covariance with transformed scores


Spline fit using 1 knots and polynomials of degree 2

Locally Weighted Regression (LWR)

- LWR finds training samples "closest" to new samples in dependent variable space and forms local model
- Points further from new sample weighted less
- LWR routine used here:
  - decomposes independent varibles with Principal Components (similar to SVD)
  - forms linear local model
  - Mahalanobis based distance measure (Euclidean based on adjusted PC scores)
  - *Weighting function* $(1 - d^3)^3$ *d furthest local point*
- Models cross-validated over:
  - number of local points
  - number of PCs retained

Identification Test System


System input (u)

Voltage to variable speed pump

Voltage from level indicator

System output (y)

Dynamic Model Forms

- Both Finite Impulse Response (FIR) and Auto-Recursive eXtensive variable (ARX) forms used
- FIR models use last six inputs:
  y(k) = *F* [u(k-1), u(k-2), ... , u(k-6)]
- ARX models use last 5 inputs and last *output*:
  y(k) = *F* [u(k-1), u(k-2), ... , u(k-5), y(k-1)]
- In non-linear systems, final steady-state *may* depend on initial condition -> ARX form required
- If system has no hysteresis, FIR form better for forecasting several points ahead

GA Model Form

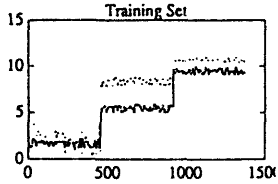
- Takes linear, squares, cubes and cross terms of ARX inputs. Selects from the following terms:
  y(k) = *C* [u(k-1) u(k-2) u(k-3) u(k-4) u(k-5) y(k-1)
  u(k-1)^2 u(k-2)^2 u(k-3)^2 u(k-4)^2 u(k-5)^2
  y(k-1)^2 u(k-1)^3 u(k-2)^3 u(k-3)^3 u(k-4)^3
  u(k-5)^3 y(k-1)^3 u(k-1)*u(k-2) u(k-1)*u(k-3)
  u(k-1)*u(k-4) u(k-1)*u(k-5) u(k-1)*y(k-1)
  u(k-2)*u(k-3) u(k-2)*u(k-4) u(k-2)*u(k-5)
  u(k-2)*y(k-1) u(k-3)*u(k-4) u(k-3)*u(k-5)
  u(k-3)*y(k-1) u(k-4)*u(k-5) u(k-4)*y(k-1)
  u(k-5)*y(k-1)]
- Over 8 billion possible model structures, hundreds of possible cross-validation sets

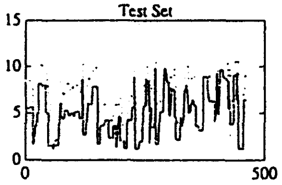
Neural Network Workshop for the Hanford Community

Identification Tests Performed

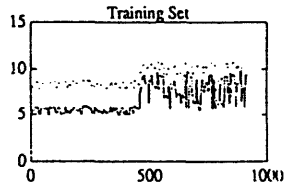
- A series of six identification tests performed
- Models identified then tested on entirely new data set

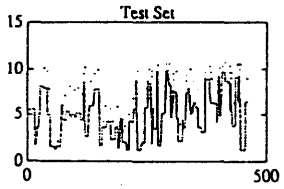
Interpolation test data





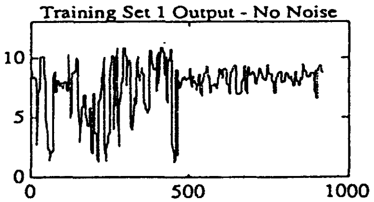
Extrapolation test data

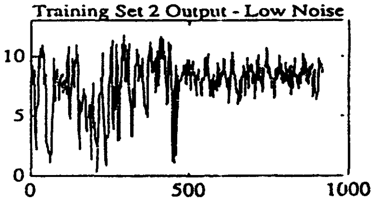


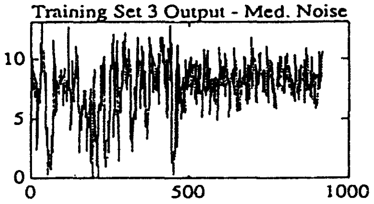


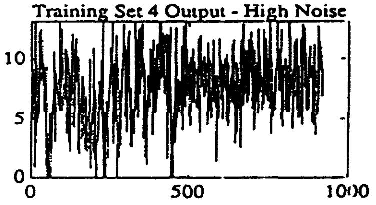
Noise Robustness Tests

- Progressively more noise added to output of identification data set
- Models identified then tested on "noise free" data

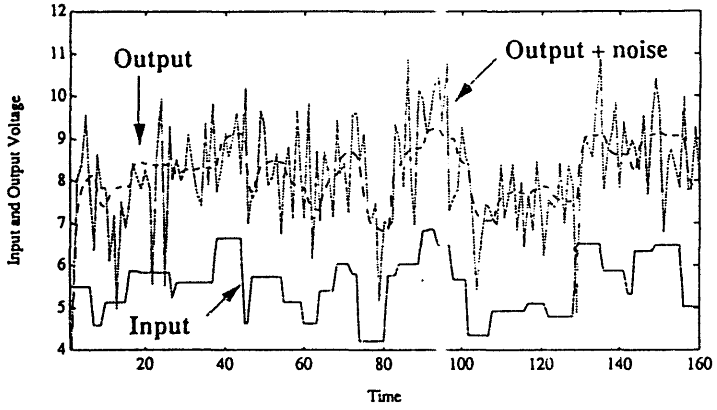




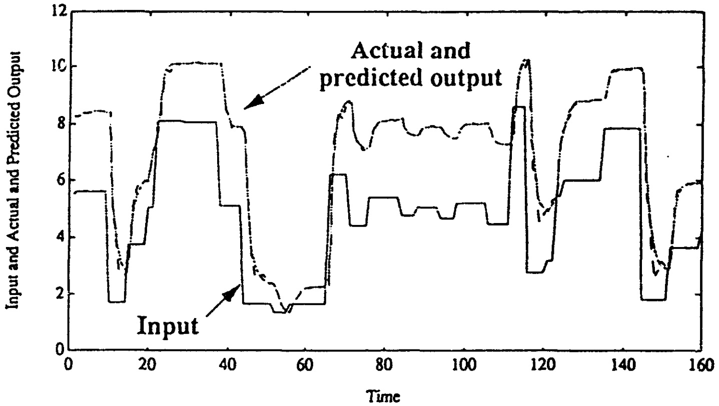




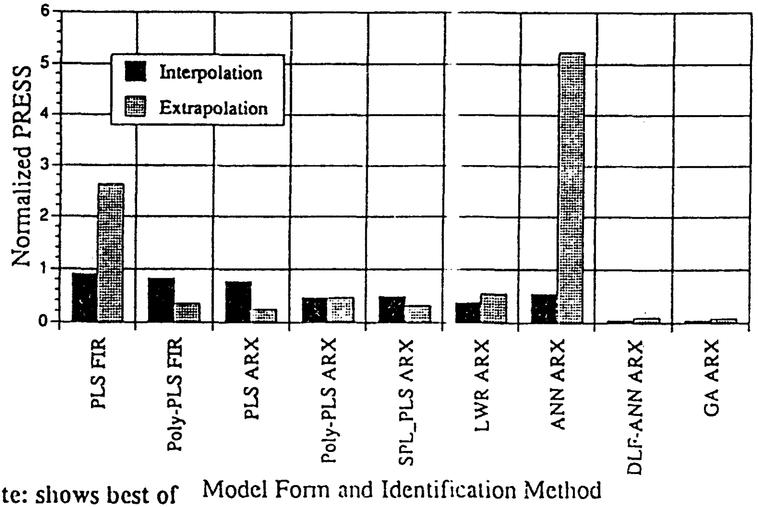
Typical Training Data Set



Typical Test Data Set

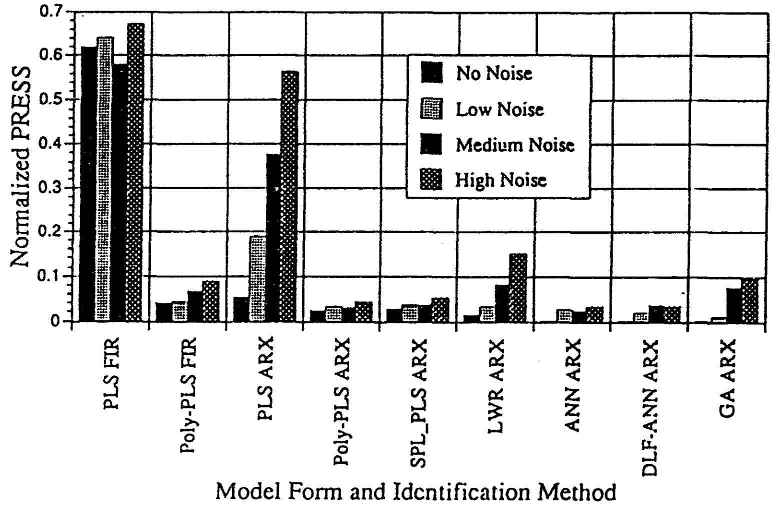


Results for Interpolation and Extrapolation



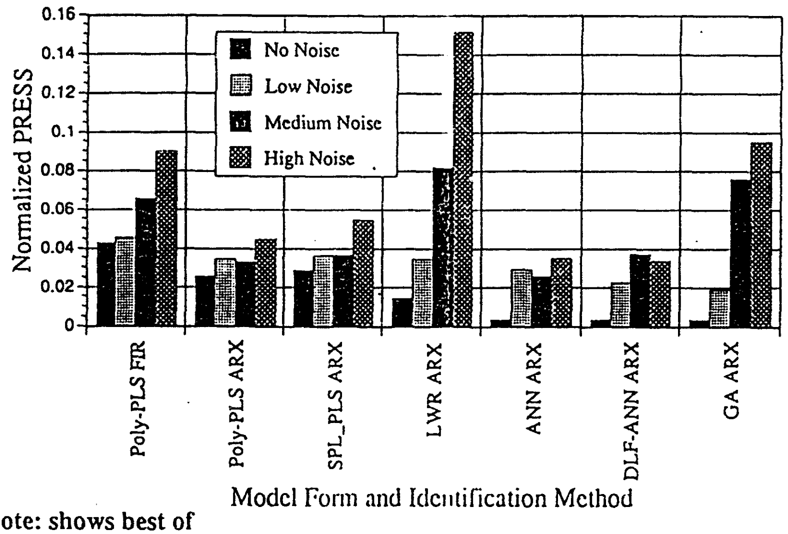
Note: shows best of
GA and ANN results

Results from Noise Tests



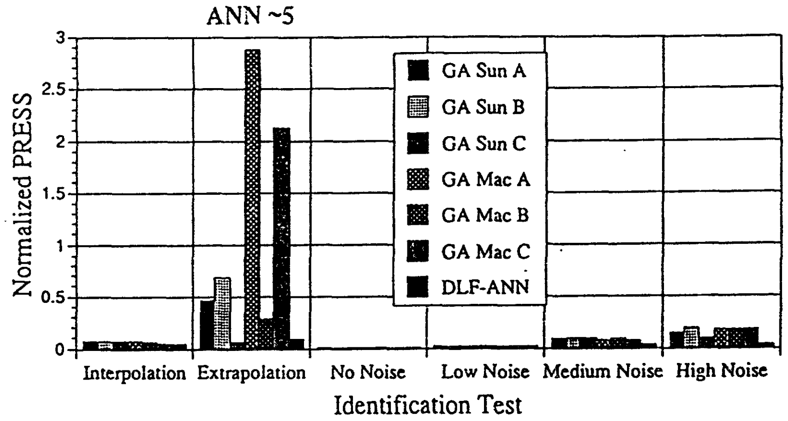
Note: shows best of
GA and ANN results

Non-Linear Methods Only



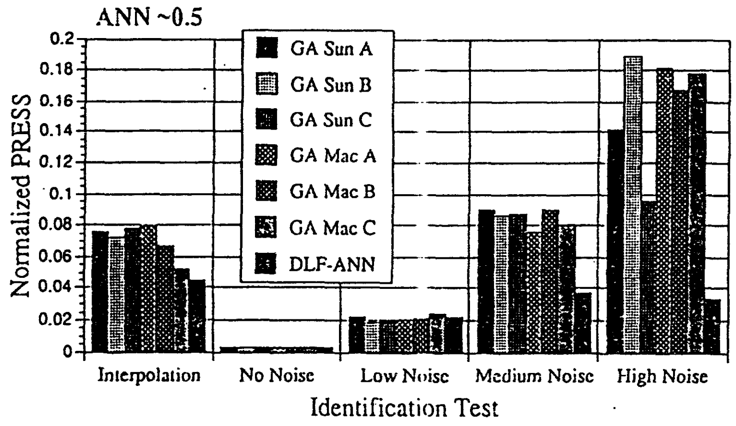
Note: shows best of
GA and ANN results

Variability in GA Solutions



Note: shows best
of ANN results

Variability in GA Solution Close Up



Note: shows best
of ANN results

Final GA Model Forms

- Terms used in "low noise" solution highlighted
  y(k) = *C* [u(k-1) u(k-2) u(k-3) u(k-4) u(k-5) y(k-1)
  u(k-1)^2 u(k-2)^2 u(k-3)^2 u(k-4)^2 u(k-5)^2
  y(k-1)^2 u(k-1)^3 u(k-2)^3 u(k-3)^3 u(k-4)^3
  u(k-5)^3 y(k-1)^3 u(k-1)*u(k-2) u(k-1)*u(k-3)
  u(k-1)*u(k-4) u(k-1)*u(k-5) u(k-1)*y(k-1)
  u(k-2)*u(k-3) u(k-2)*u(k-4) u(k-2)*u(k-5)
  u(k-2)*y(k-1) u(k-3)*u(k-4) u(k-3)*u(k-5)
  u(k-3)*y(k-1) u(k-4)*u(k-5) u(k-4)*y(k-1)
  u(k-5)*y(k-1)]
- 20 out of 33 terms used
- Solution quite consistent from run to run
- Selected terms not what I would have guessed!!

Results

- ARX models more effective at one-step ahead predictions, as expected
- DLF-ANN model structure appropriate for this type of problem, most consistently good results
- GA works very well when test and training sets similar, but results can be highly variable when they are not
- Note: GA selects model structure
- Poly-PLS generally "acceptable," very fast
- Poly-PLS models better than SPL_PLS in all cases
- LWR quite affected by noise, result of subset selection?

Conclusions

- Model structure is critical
- If you've got the time and computer power, DLF-ANNs good choice for this problem
- Good promise in genetic algorithm, some very good results and computationally efficient
- Use Poly-PLS for very fast, generally acceptable results

Concerning Neural Networks Marvin Minsky Warns:

"They write a paper saying,
'Look it did this,' and they don't say,
'Look, it can't do that.'
Most of them are not doing good science,
because they're hiding the deficiencies."

Scientific American, November 1993

Common "Deficiencies"

- Lack of an appropriate "benchmark" from other accepted approach
- Showing the best of many solutions
- Confusing fit and prediction
- Little discussion of effect of "meta-parameters"
- No account of similar approaches that failed

Combining Neural Networks

Sherif Hashem

Pacific Northwest Laboratory, Molecular Science Research Center,
Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-6995 fax: (509) 375-6631 internet: s_hashem@pnl.gov

Neural network based modeling often involves trying multiple networks with different architectures and/or training parameters in order to achieve acceptable model accuracy. Typically one of the trained networks is chosen as best while the rest are discarded. In this talk, we discuss the use of the optimal linear combinations of a number of trained networks instead of only using the best network. Optimal linear combinations can also be applied to combined neural network and non-neural network models.

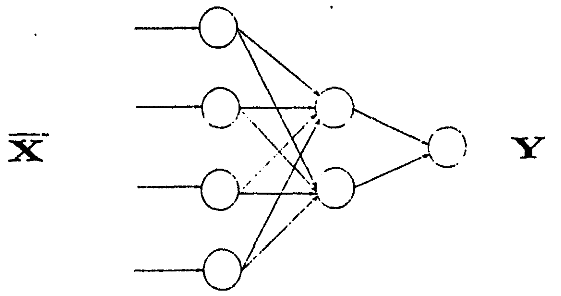
OUTLINE

- Model Construction.
- Combining Models.
- Optimal Linear Combinations of Neural Networks.
  - Definition.
  - Combination weights.
- Benefits of Combining:
  - For well-trained networks.
  - For poorly trained networks.
- Ill Effects of Collinearity.
- Concluding Remarks.

MODEL CONSTRUCTION

- **Modeling problem:** Given a (process) data set, construct a model that "closely" approximates the underlying process.
- Modeling involves:
  - Searching for the "true" model.
  - Computing (estimating) the model parameters.
- As a result, a number of estimated models are often constructed. Typically, one model is chosen as best, while the rest are discarded.

MODEL CONSTRUCTION



Neural Network Model

$$Y = f_{NN}(\vec{X})$$

Modeling involves selecting:

- A network topology.
- A learning technique.

COMBINING MODELS

- Simple (unweighted) averaging: Laplace 1818.

$$\text{Given } Y_1, \ldots, Y_p, \quad \text{use } \widetilde{Y} = \sum_{i=1}^{p} Y_i/p.$$

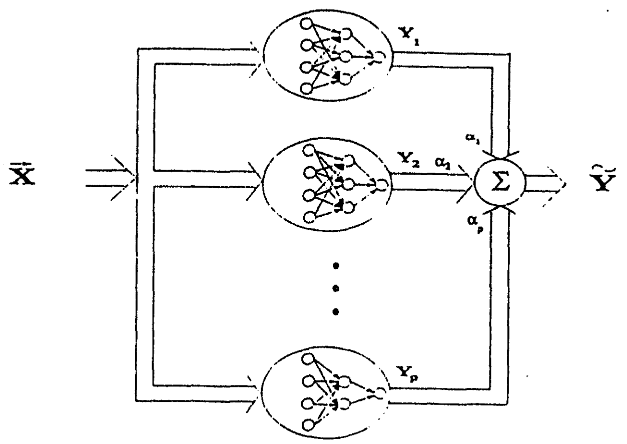
- Weighted averaging: Bates and Granger 1969, and Ried 1969.

$$\text{Given } Y_1, \ldots, Y_p, \quad \text{use } \widetilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i, \quad \text{where } \sum_i \alpha_i = 1.$$

- Optimal linear combinations: Granger and Ramanathan 1984.

$$\text{Given } Y_1, \ldots, Y_p, \quad \text{use } \widetilde{Y} = \alpha_0 + \sum_{i=1}^{p} \alpha_i Y_i.$$

OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS



OLC of Neural Networks

$$\widetilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}.$$

- **Current approach:** Train many NNs, then pick the "best."
- **New approach:** Optimal Linear Combinations (OLC) of NNs.

COMBINATION-WEIGHTS

Optimality Criterion:

Minimize the Mean Squared Error (MSE) over observed data.

$$\text{MSE} = \text{E}_{\vec{X}}(r(\vec{X}) - \widetilde{Y})^2.$$

The MSE-optimal weights:

$$\vec{\alpha}^* = \Phi^{-1}\vec{\Theta},$$

where

$\Phi = [\phi_{ij}] = [\text{E}(y_i(\vec{X})\, y_j(\vec{X}))]_{p\times p}$ and $\vec{\Theta} = [\theta_i] = [\text{E}(r(\vec{X})\, y_i(\vec{X}))]_{p\times 1}$.

In practice: Given a data set $\mathcal{D}$, estimate $\vec{\alpha}^*$ using

$$\hat{\phi}_{ij} = \sum_{k=1}^{|\mathcal{D}|} (y_i(x_k)\, y_j(x_k))/|\mathcal{D}| \quad \forall\, i, j\,;$$

$$\hat{\theta}_i = \sum_{k=1}^{|\mathcal{D}|} (r(x_k)\, y_i(x_k))/|\mathcal{D}| \quad \forall\, i\,.$$

OTHER FORMS OF MSE-OLC

A. Unconstrained MSE-OLC with a constant term.

$$\widetilde{Y} = \sum_{i=0}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}.$$

B. Constrained MSE-OLC with a constant term.

$$\widetilde{Y} = \sum_{i=0}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}, \quad \sum_{i=1}^{p} \alpha_i = 1.$$

C. Constrained MSE-OLC without a constant term.

$$\widetilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}, \quad \sum_{i=1}^{p} \alpha_i = 1.$$

D. Convex MSE-OLCs:

$$\widetilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}, \quad \sum_{i=1}^{p} \alpha_i = 1, \quad 1 \geq \alpha_i \geq 0.$$

BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 1

- **Problem:** Consider approximating

$$r(X) = \sin[2\,\pi\,(1 - X)^2], \quad \text{where} X \in [0, 1].$$

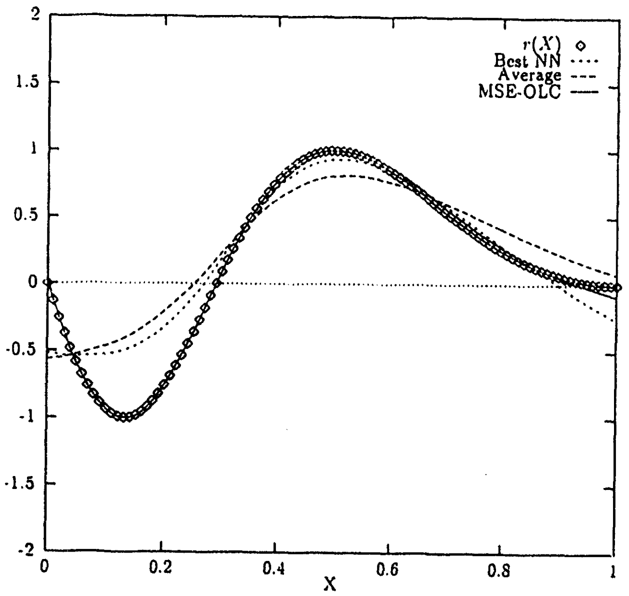
- NN model:
  - *Topology:* Two 1–3–1, two 1–2–2–1, & two 1–4–1 NNs.
  - *Training algorithm:* Using Error Backprop for 5000 iterations.
  - *Training data:* 10 uniformly distributed data points.
  - *MSE-OLC fitting data:* same 10 points.
- Resultant *true* MSE:
  - *Best NN (NN6):* 0.044.
  - *Simple averaging:* 0.072.
  - *MSE-OLC:* 0.00020;

  99+ % less than NN6 or simple averaging.

BENEFITS OF COMBINING

FOR WELL-TRAINED NETWORKS

EXAMPLE 1 (Cont.)



BENEFITS OF COMBINING

FOR POORLY TRAINED NETWORKS

EXAMPLE 2

- **Problem:** Consider approximating

$$r(X) = \sin[2\,\pi\,(1 - X)^2], \quad \text{where} X \in [0, 1].$$

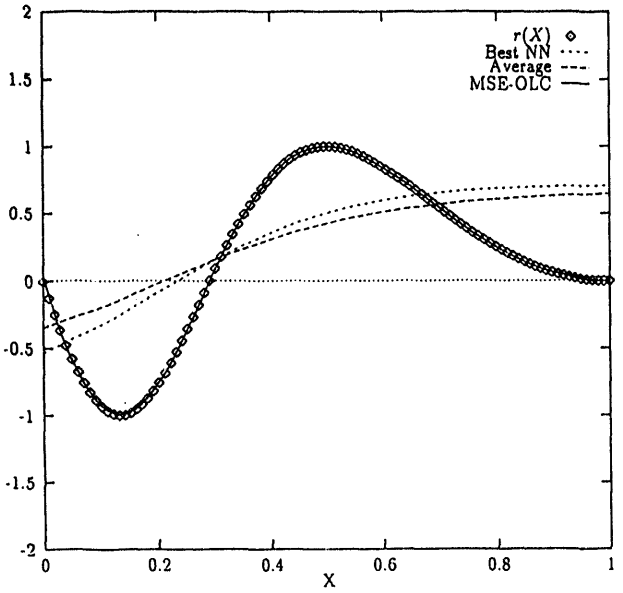
- NN model:
  - *Topology:* Two 1–3–1, two 1–2–2–1, & two 1–4–1 NNs.
  - *Training algorithm:* Using Error Backprop for 2000 iterations.
  - *Training data:* 10 uniformly distributed data points.
  - *MSE-OLC fitting data:* same 10 points.
- Resultant *true* MSE:
  - *Best NN (NN6):* 0.219.
  - *Simple averaging:* 0.241.
  - *MSE-OLC:* 0.000060;

  99+ % less than NN6 or simple averaging.

BENEFITS OF COMBINING

FOR POORLY TRAINED NETWORKS

EXAMPLE 2 (Cont.)



CONCLUDING REMARKS

- MSE-OLCs can significantly improve model accuracy.
- The effectiveness of MSE-OLC is not dependent on the accuracy of the component networks.
- MSE-OLC is straightforward and requires modest computational effort.
- MSE-OLC can be used to create hybrid models containing non-neural network components.

ILL EFFECTS OF COLLINEARITY

MSE-OLCs:

$$\tilde{Y} = \sum_{i=1}^{p} \alpha_i Y_i = \vec{\alpha}^t \vec{Y}.$$

Unconstrained MSE-OLC Weights:

$$\vec{\alpha}^* = \Phi^{-1}\vec{\Theta},$$

where

$\Phi = [\phi_{ij}] = [\mathrm{E}(y_i(\vec{X})y_j(\vec{X}))]_{p\times p}$ and $\vec{\Theta} = [\theta_i] = [\mathrm{E}(r(\vec{X})y_i(\vec{X}))]_{p\times 1}$.

Constrained MSE-OLC Weights:

$$\vec{\alpha}^* = \Omega^{-1}\vec{1}\,/\,(\vec{1}^t\,\Omega^{-1}\,\vec{1})\,,$$

where $\Omega = [\omega_{ij}] = \left[\mathrm{E}\left(\delta_i(\vec{X})\,\delta_j(\vec{X})\right)\right]$ is a $p \times p$ matrix, and $\vec{1}$ is a $p \times 1$ vector with all components equal to one.

Computational Ill Effects:

Near singular matrices (inversion, sensitivity, round-off errors).

Statistical Ill Effects:

Collinearity can undermine the robustness (generalization ability) of the MSE-OLC.

REFERENCES

J. M. Bates and C. W. J. Granger The combination of forecasts. *Operational Research Quarterly*, 20(4):451–468, 1969.

D. A. Belsley. *Conditioning Diagnostics: Collinearity and Weak Data in Regression.* John Wiley & Sons, New York, 1991.

R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5:559–583, 1989.

C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*. 1(1):114–148, 1986.

C. W. J. Granger and R. Ramanathan Improved methods of combining forecasts. *Journal of Forecasting*, 3:197–204, 1984.

S. Hashem. *Optimal Linear Combinations of Neural Networks.* PhD thesis, School of Industrial Engineering, Purdue University, Dec. 1993.

P. d. Laplace *Deuxième Supplément a la Théorie Analytique des Probabilités.* Courcier, Paris, 1818. Reprinted (1847) in *Oeuvers Complètes de Laplace,* Vol. 7 (Paris, Gauthier-Villars), 531–580.

D. J. Ried *A Comparative Study of Time Series Prediction Techniques on Economic Data.* PhD thesis, University of Nottingham, Nottingham, UK, 1969

Parallel and Distributed Gradient Decent Learning

I. Mehr, Z. Obradović, R. Venkateswaran

Washington State University, School of Electrical Engineering and Computer Science
Pullman, WA 99164-2752
phone: (509) 335-6601 fax: (509) 335-3818 internet: zoran@eecs.wsu.edu

First part of the talk describes a distributed learning algorithm which uses cooperative efforts of several identical neural networks for more efficient gradient descent learning (join work with R. Venkateswaran). In contrast to the sequential gradient descent, in this algorithm it is easy to select learning rates such that the number of epochs for convergence is minimized. It has been implemented on a network of heterogeneous workstations using p4. Results are presented where few learners cooperate and learn much faster than if they learn individually.

Second part of the talk describes a highly parallel learning approach based on repetitive bounded depth trajectory branching (joint work with I. Mehr). This algorithm has objectives of improving generalization and speeding up convergence by exploring a number of alternative trajectories in parallel in order to find one that avoids local minima. The experimental results show an improved generalization compared to the standard back-propagation learning algorithm.

Contents

- **Distributed Learning**
  Venkateswaran, R. and Obradović, Z. (under review) "Efficient Learning through Cooperation." *1994 World Congress on Neural Networks*, San Diego, CA.

- **Highly Parallel Learning**
  Mehr, I., and Obradović, Z., (1994) "Parallel Neural Network Learning Through Repetitive Bounded Depth Trajectory Branching," *Proc. IEEE 8th Int. Parallel Processing Symposium*, Cancun, Mexico.

Drawbacks of Back-Propagation:

- Slow Learning
- Local Minima Problem
- Learning Rate Determination

Objective: To speed-up learning and improve generalization through cooperative efforts of several identical neural networks.

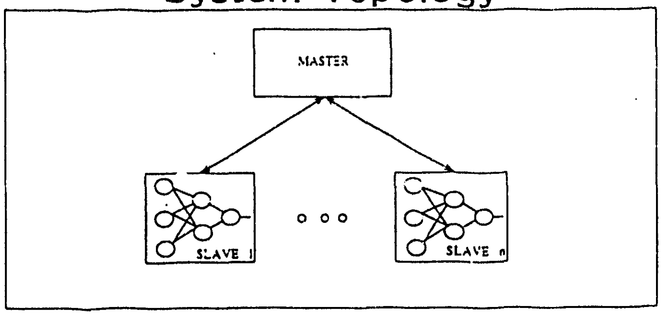
Cooperation for Distributed Learning

Venkateswaran, R. and Obradović, Z. (under review) "Efficient Learning through Cooperation." *1994 World Congress on Neural Networks*, San Diego, CA.

Cooperative Learning:

- Several **slave** processes run the standard back-propagation algorithm concurrently controlled by the **master** process.
- All **slave** processes work on neural networks of identical topology, each using a local copy of the training set.

System Topology



The Algorithm:

1. Master initializes weights.
2. Master broadcasts weights to slaves.
3. Slaves adjust the weights using BP. Each slave uses its own learning rate (different from others).
4. Periodically, slaves cooperate.

- The communication graph is simple.
- The number of communications is small.
- Suitable for a distributed implementation.

Pattern Classification Results

Problem:	To classify 'A', 'I' and 'O'.		
Dimension:	2	Classes:	3
Architecture :	2-9-3	Examples:	16
Learned :	100%	Era(epochs):	50

One Node

0.01	0.05	0.1	0.125	0.15	0.175
8708	1819	1295	1179	1419	>10000

Two Nodes Cooperation

0.05, 0.15	0.1, 0.15	0.01, 0.15	0.05, 0.175
1099	1040	1374	985

Three Nodes Cooperation

0.05, 0.10, 0.15	0.01, 0.1, 0.2
1149	1148

Four Nodes Cooperation

0.05, 0.1, 0.15, 0.2
946

Cooperation Models

- **Epoch-based Cooperation**
  Slaves cooperate after a specified number of epochs (one *era*). The master forms a new hypothesis after each era by averaging these weights.
- **Time-based Cooperation**
  The era is specified as a duration of time. If all slaves run for the same duration, no machine will be idle.
- **Dynamic Rates Cooperation**
  Initial learning rates spread uniformly in (0,1) range. After few eras, the learning rates range is reduced around the currently best rate.

Two-Spirals Epoch-based Results

Problem :	Two-Spirals Problem.		
Dimension :	2	Classes:	2
Architecture :	2-5-1	Examples:	40
Learned :	100%	Era(epochs):	100

One Node

0.05	0.1	0.15	0.2	0.25	0.3	0.35
>30000	8799	3693	2593	2238	1727	>30000

Two Nodes Cooperation

0.05, 0.35	0.15, 0.35	0.25, 0.35	0.05, 0.5
2691	2122	1777	1920

Three Nodes Cooperation

0.1, 0.3, 0.5	0.15, 0.25, 0.35
1775	2103

Four Nodes Cooperation

0.1, 0.2, 0.3, 0.4
2498

Homogeneous vs Heterogeneous

Problem :	Two-Spirals Problem.		
Dimension :	2	Classes :	2
Architecture :	2-5-1	Examples:	40
Learned :	100%	Era(msec):	400

(DEC5000, DEC5000) System

0.05, 0.35	0.15, 0.35	0.25, 0.35	0.05, 0.5
40	34	28	27

(DEC5000, HP9000/735) System

0.05, 0.35	0.15, 0.35	0.25, 0.35	0.05, 0.5
6	6	6	5

(HP9000/735, DEC5000) System

0.05, 0.35	0.15, 0.35	0.25, 0.35	0.05, 0.5
22	6	7	21

Branching for Highly Parallel Learning

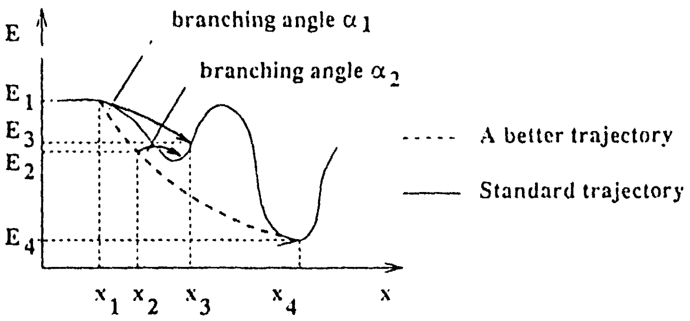
Mehr, I., and Obradović, Z., (1994) "Parallel Neural Network Learning Through Repetitive Bounded Depth Trajectory Branching," *Proc. IEEE 8th Int. Parallel Processing Symposium*, Cancun, Mexico.

Analysis of Experimental Results

- Cooperative approach is better.
- Select η_1 and η_2 such that $\eta_1 < \eta_{min} < \eta_2$
- For a heterogeneous system time based cooperation is better.
- Larger rate should be assigned to faster machine.

Objective:

Explore a number of learning trajectories in parallel in order to find one that avoids local minima.


E
branching angle α_1
branching angle α_2
E_1
E_3
E_2
E_4
x_1
x_2
x_3
x_4
x
----- A better trajectory
—— Standard trajectory

The General Branching Algorithm

(1) Initially follow a single learning trajectory corresponding to the standard algorithm.

(2) Generate new branching points after a specified number of !earning steps.

(3) At each new branching point start exploring new learning trajectories in addition to the existing ones.

(4) Continue with step 2 until B trajectories are generated.

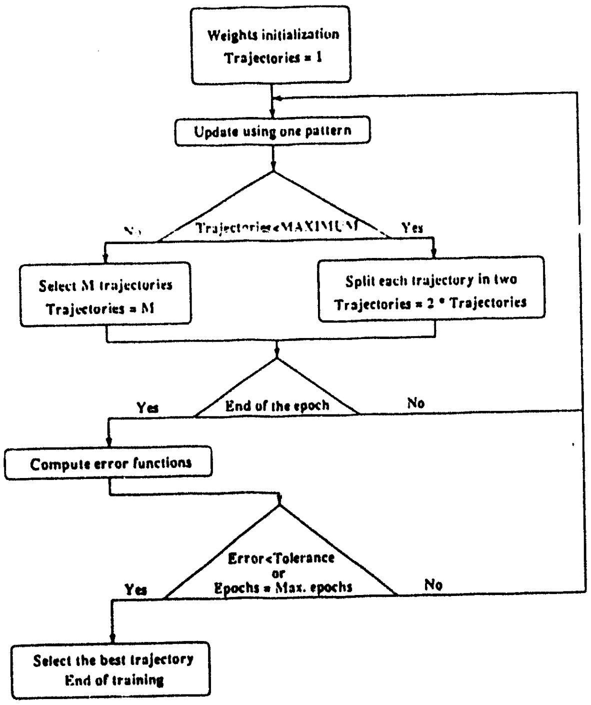
(5) Using a cross-validation test on all existing trajectories select M of those and abandon the remainder.

(6) Starting from M selected trajectories of step 5 continue constructing and exploring new trajectories on new branching points until B trajectories are constructed.

(7) Continue with step 5 until the training error is within a prespecified tolerance or for a prespecified number of steps.

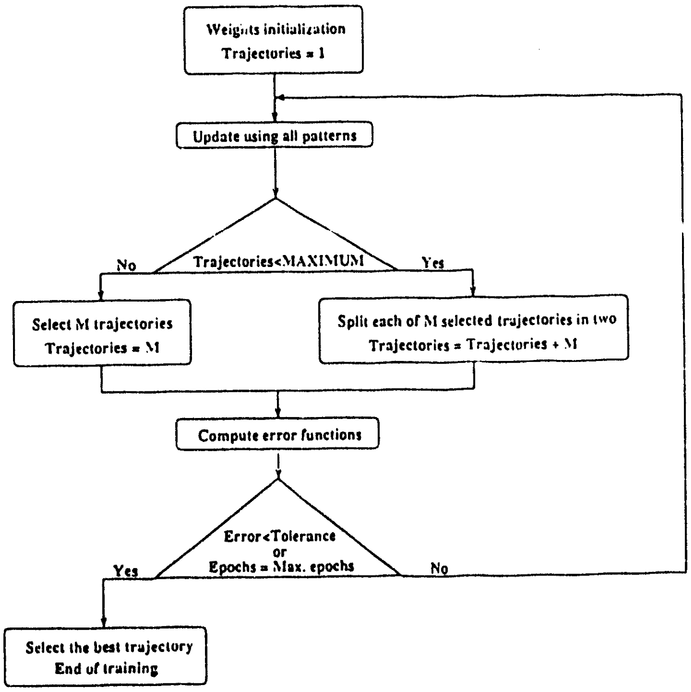
Short Term Branching Algorithm

The idea: Generate new branching points after presentation of each training pattern.



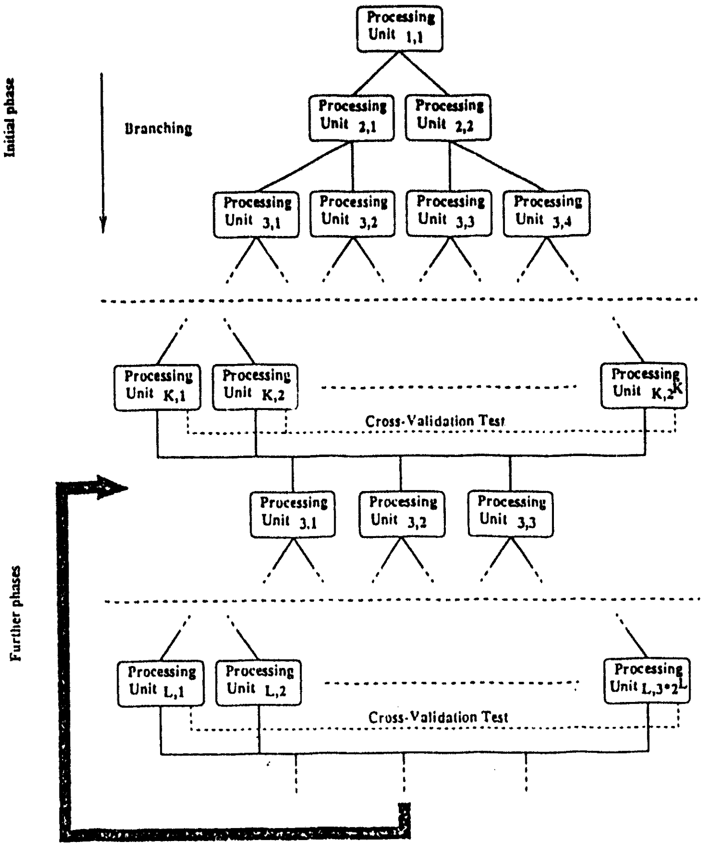
Long Term Branching Algorithm

The idea: Generate a new trajectory at the end of each epoch.



Short Term Branching Parallelization

- Branching by copying the modified network parameters to a new processor.
- The process repeats for K steps, when 2^K processors contain 2^K different networks.
- Processors performs in parallel a cross-validation evaluating their local networks.
- After selection of M of those networks, the algorithm continues.



Short Term Algorithm Analysis

A sequential implementation is very expensive

$$\frac{T_{s,s}}{T_s} \approx \frac{1}{\left\lceil log_2 \frac{P}{3} \right\rceil} \left[(B + \frac{B}{2} + \cdots + 3) + \frac{t_v B + t_c}{c_p} \right]$$

A parallel algorithm is more suitable for a highly parallel system rather that distributed system implementation since

$$\frac{T_{p,s}}{T_s} \approx 1 + \frac{t_t}{c_p} + \frac{t_c + t_v}{c_p \left\lceil log_2 \frac{P}{3} \right\rceil}$$

Here:

P - the number of available processors;

c_p - the net update time by standard alg;

t_t - a network parameters transfer time;

t_v - a single C-V test time;

t_c - branches selection time after a C-V step.

Long Term Algorithm Analysis

A sequential is still very expensive

$$\frac{T_{s,l}}{T_l} \approx \frac{B+3}{2} + \frac{3t_v}{c_e} + \frac{3t_c}{Bc_e}$$

Parallel implementation is time efficient since

$$\frac{T_{p,l}}{T_l} \approx 1 + \frac{t_t}{c_e} + \frac{3\,(t_c + t_v)}{Bc_e}$$

Here:

c_e - the computing between two updates by the standard algorithm.

Trajectory Construction Methods

A new trajectory

$$W + \Delta W^* = [w_1 + b_1, \ldots, w_p + b_p]$$

is constructed using one of following methods:

1. Compute $b_i = 0$ if $i = j$ and $b_i = a_i$ otherwise, where j is given by $a_j = min\{a_i\}$ and $\{a_i\}$ are the standard BP updates.

2. Compute $b_i = 0$ if $i = j$ and $b_i = a_i$ otherwise, where j is given by $a_j = max\{a_i\}$.

3. $[b_1, \cdots, b_p] = [0, \cdots, 0, a_{k+1}, \cdots, a_p]$ where k is the largest integer such that the angle between ΔW and ΔW^* is less than 45°.

4. $[b_1, \cdots, b_p] = [a_1, \cdots, a_{j-1}, 2a_j, a_{j+1}, \cdots, a_p]$, where j is given by $a_j = max\{a_i\}$.

5. Compute $b_i = -\eta^* \frac{\partial E}{\partial w_i}$, $\eta^* \neq \eta$, where η and η^* are the learning rates for the standard back-propagation trajectory and for the new trajectory.

Methods Comparison

Algorithm	Run time	Error
Back-propagation	45 min.	10%
Method 1	45 min.	46%
Method 2	45 min.	40%
Method 3	45 min.	44%
Method 4	45 min.	40%
Method 5	45 min.	45%

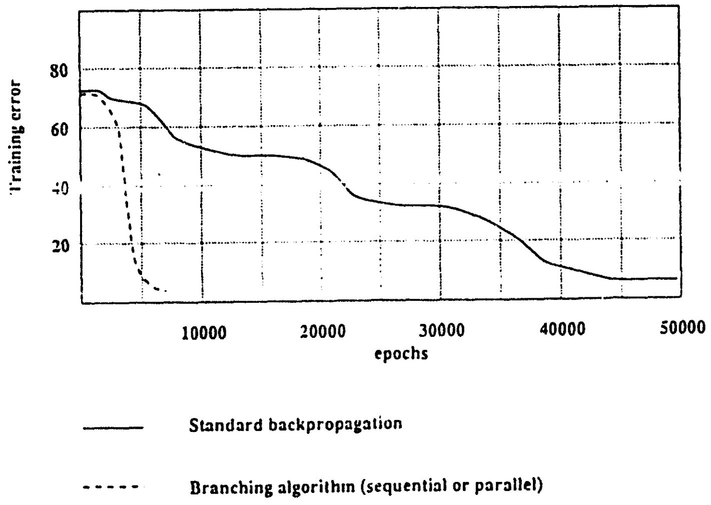
Standard Back-Propagation

Data	Spirals	Breast Cancer
Configuration	2-6-1	10-10-1
Learning rate	0.05	0.05
Number of patterns	100	250
Training error	21%	4%
Generalization	73%	76%
Epochs	50000	25000
Run time	45 min.	65 min.

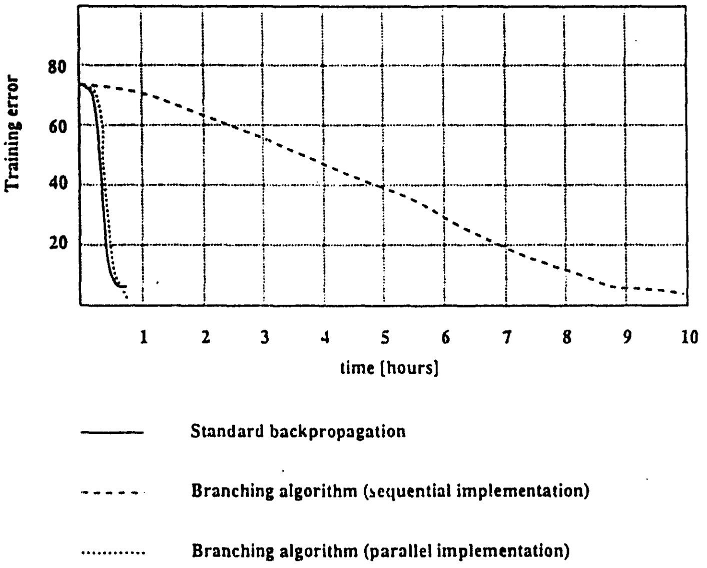
Short Term Branching Algorithm

Data	Spirals	Breast Cancer
Configuration	2-6-1	10-10-1
Learning rate	0.1	0.4
Number of patterns	100	250
Training error	18%	3.5%
Generalization	77%	78%
Epochs	7000	5000
Sequential run time	14 hours	24 hours
Parallel run time	47 min.	63 min.

Training Epochs Comparison



Computing Time Comparison



Analysis of Experimental Results

- Branching improves the back-propagation generalization.
- Parallelization improves the branching algorithm efficiency.
- The branching idea is applicable to other learning algorithms.

Conclusions

Both approaches (distributed and highly parallel) show:

- improved efficiency
- improved generalization.

Neural Network Algorithms for VLSI Design and Test Automation

Dr. Jack Meador
Washington State University, School of Electrical Engineering and Computer Science
Pullman, WA 99164-2752
phone: (509) 335-5363 fax: (509) 335-3818 internet: meador@eecs.wsu.edu

The complex nature of VLSI design has for some time demanded the use of computational tools which can assist with important tasks such as cell placement and wire routing. The development of testing algorithms for manufactured ICs is also typically too complex for an unassisted designer to tackle. This presentation addresses the experimental application of existing and new neural network algorithms to basic problems in VLSI design and automation. Although VLSI applications are the prime focus here, the extension of the basic concepts to other application domains will also become evident.

In the most general sense, IC test involves the determination of circuit functionality based upon an observed response to a stimulus. The value of IC test has been established for many years in the digital circuit domain. Digital IC test design focuses primarily upon the determination of a stimulus set which adequately exercises all internal portions of a circuit. Any deviation whatsoever from a corresponding set of output responses signifies a malfunction. The design of an adequate stimulus can be cast as a discrete optimization problem which has been approached in a variety of ways including the use of Hopfield neural networks.

Once a stimulus is determined for a digital system, testing can proceed relatively independent of parametric variation in the device fabrication process since digital circuit function is inherently immune to such variation. Analog and mixed-signal IC test does not enjoy a similar luxury. The fundamental nature of analog circuitry and its sensitivity to random process variation makes the test problem somewhat more complex. It becomes necessary to cast analog and mixed-signal IC test as a statistical pattern recognition problem. One approach to solving this problem is to use a trainable pattern classifier which minimizes mean square error for a given training set. Feedforward neural network classifiers naturally fit this kind of problem. One difficulty associated with feedforward network training for this task involves the need to execute a large number of accurate circuit simulations. This first part of the presentation will briefly introduce a technique for reducing the number of simulations needed while accelerating network training and retaining network accuracy.

The design of a digital integrated circuit also typically involves the determination of the placement and interconnection of a vast number of small components. Although usually possible. it is typically impractical for designers to determine all aspects of component placements and interconnections by hand given typical engineering cost constraints. Because of this, algorithms for automatically determining a large percentage of component placements and interconnections are used to help offload a substantial portion of the task onto engineering workstations.

With increased IC size and density comes greater system complexity. Existing placement and routing algorithms thus become increasingly constrained by their inherent sequential nature. Neural network algorithms organized to solve such problems provide one approach for parallel solutions. The second part of this presentation will provide an introduction to the application of a new variation of the Kohonen feature map algorithm to the parallel solution of these and related optimization problems.

Overview

VLSI Test

problem statement

prior work

simulation sampling for neural diagnostic training

VLSI Cell Placement and Routing

problem statement

prior work

elastic nets for maze routing

Digital v.s. Analog Test

Digital Test

"stuck-at" fault models

test pattern synthesis

high fault coverage

Analog/Mixed-Signal Test

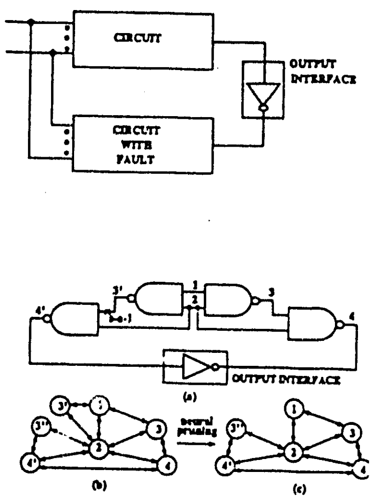
catastrophic and parametric fault models

classifier synthesis

low classification error

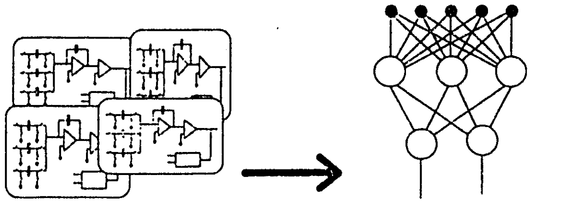
Digital Test Pattern Synthesis

[Chakradhar 91]



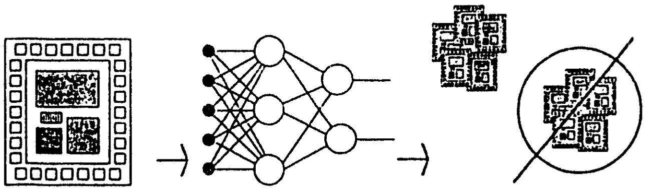
Analog/Mixed-Signal Test

Fault Classifier Design



Simulation Sample   Trainable Pattern Classifier

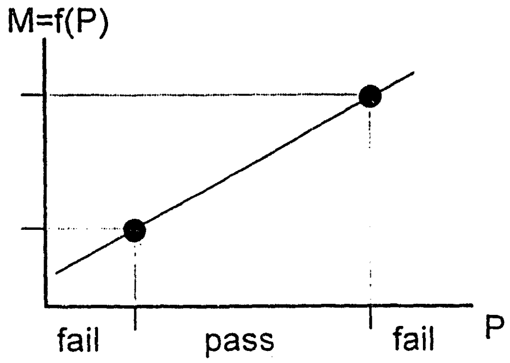
Production Test



IC Measurements   Trained Classifier   Fault Discrimination

Hidden Parameter Prediction

[Starzyk 90]



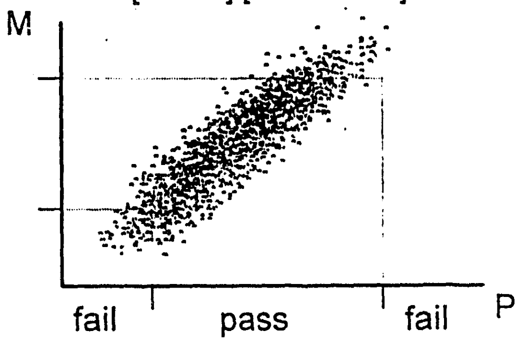
FFN computes an approximation of f $^{-1}$

Training set derived from all combinations of worst-case acceptable parameter deviations

Training set does not capture fab process statistics

Statistical Pattern Recognition

[Lin 90] [Meador 91]



FFN approximates a Bayes statistical decision

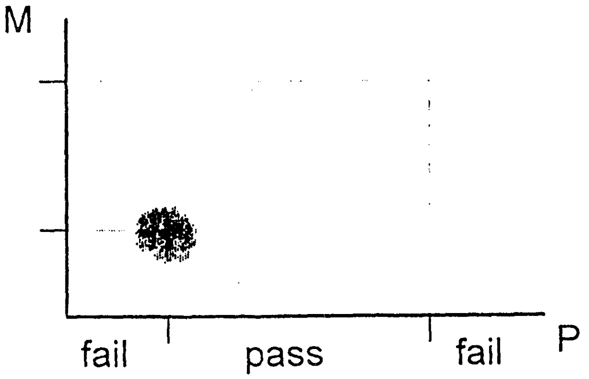
Training set derived from Monte Carlo simulation of fab process statistics

FFN consistently performed as well or better than traditional classifiers, also requires much fewer FLOPS to perform a classification

Computationally expensive to generate training set, also expensive to train FFN

Boundary Sampling

[Wu 94]



FFN approximates a Bayes statistical decision

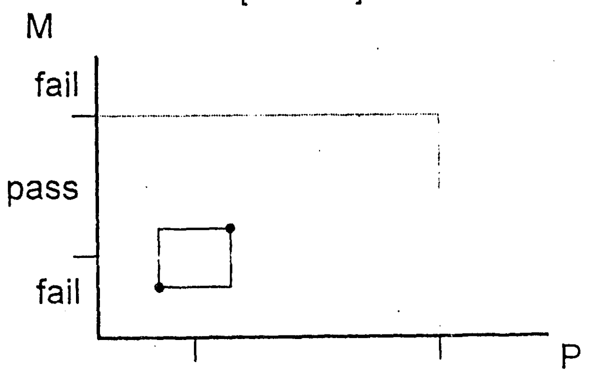
Training set statistics biased toward important boundaries

More efficient use of simulation time

Faster network training

Boundary Sampling with Performance Based Faults

[Wu 94a]



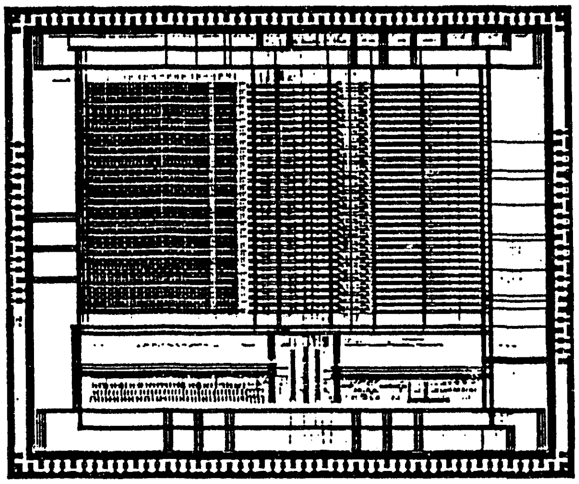
FFN approximates a Bayes statistical decision

Training set developed using orthogonal arrays centered on important boundaries

Fault definitions are in measurement space

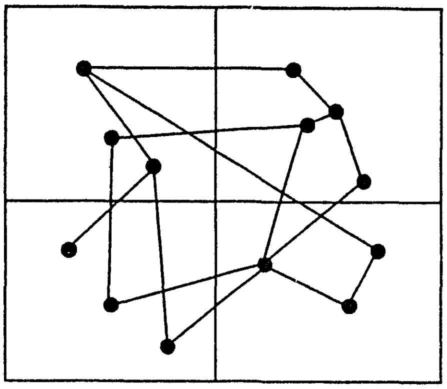
Important boundaries in parameter space are determined by low-order response surface modeling

Cell Placement and Wire Routing in VLSI



Hopfield Network Based Cell Placement

[Sriram 90]

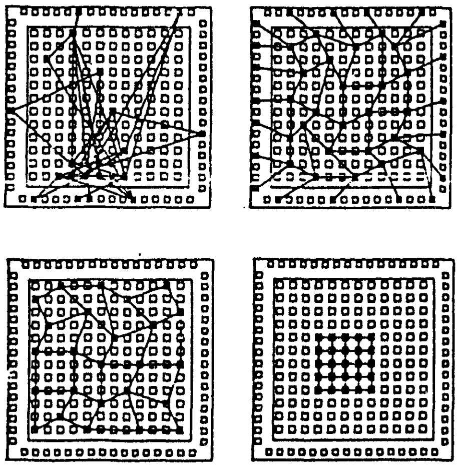


approximate solutions to min-cut placement via hierarchical graph quadrisectioning

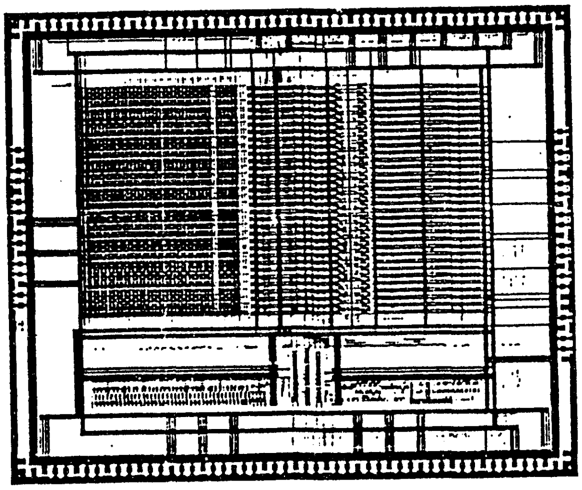
requires O(n2) connections for n modules

Kohonen Feature Map Based Cell Placement

[Hemani 90][Shen 92][Blight 92]



Maze Routing

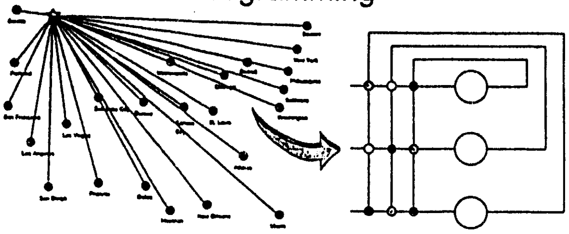


two parts:
traveling salesman
shortest path with obstacles

Hopfield Network approach to the TSP

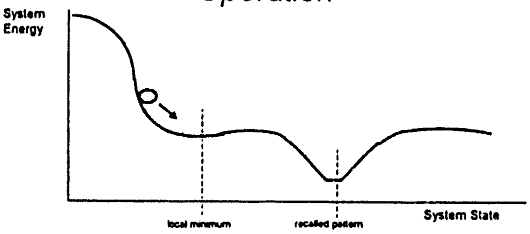
[Hopfield 85]

Programming



N^2 distances between cities

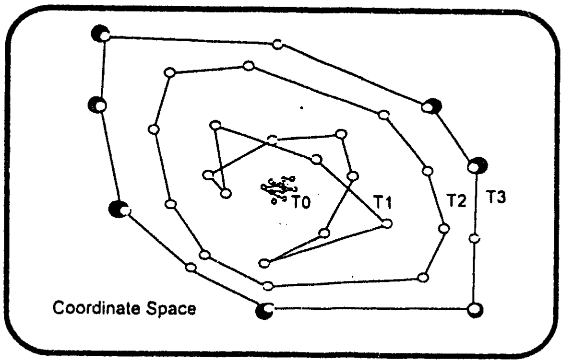
Operation



Network size grows with N2

"Elastic Net" approach to the TSP

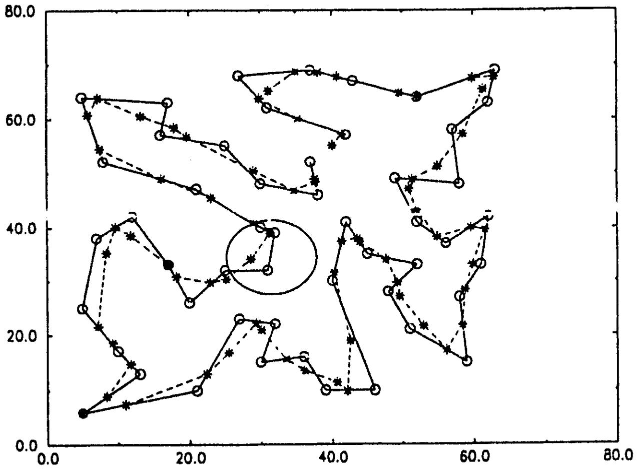
[Durbin 87]



Constrained unsupervised adaptation

"Stochastic Elastic" approach to the TSP

[Meador 93]

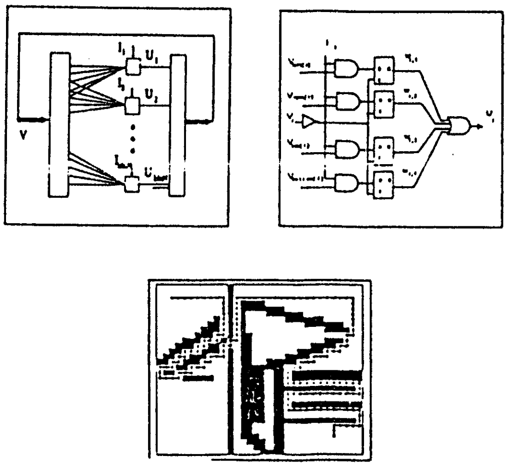


1-D Kohonen feature map -- network size proportional to number of cities

fixed local interconnections for simplified VLSI implementation

Hopfield Network approach to Path Planning

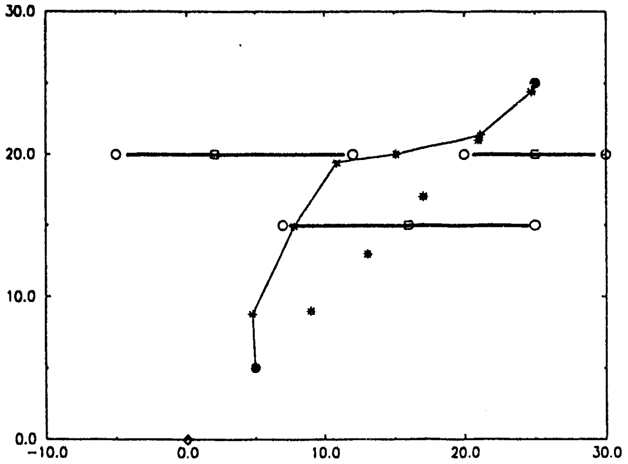
[Chan 93]



Network size proportional to grid area

Elastic Net approach to Path Planning

[Meador ??]



Network size proportional to number of "bends" needed

Summary

Digital test pattern synthesis using Hopfield networks

Hidden parameter prediction for analog circuit test using feedforward networks

Statistical pattern recognition for analog circuit test using feedforward networks

Hopfield Network Based Cell Placement

Kohonen Feature Map Based Cell Placement

Hopfield Networks and Kohonen Feature Maps for attacking wire routing problems

References

[Blight 92] Blight, D. and McLeod, R., "Self-Organizing Kohonen Maps for FPGA Placement," in **Field Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping**, Springer-Verlag, 1992.

[Chakradhar 91] Chakradhar, S.T., V.D. Agrawal, and M.L. Bushnell, **Neural Models and Algorithms for Digital Testing**, Kluwer Academic Publishers, 1991.

[Chan 93] Chan, H.T., Tam, K.S., and Leung, N.K., "A Neural Network Approach for Solving the Path Planning Problem," IEEE Int. Symp. on Circuits and Systems, pp. 2454-2457, 1993.

[Durbin 87] Durbin, R. and D. Willshaw, "An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method," **Nature** 326, 689-691, 1987.

[Hemani 90] Hemani, A., and A. Postula, "Cell Placement by Self-Organization," Neural Networks, Vol. 3, pp. 377-383, 1990.

[Hopfield 85] Hopfield, J.J., and D.W. Tank, "Neural Computation of Decisions in Optimization Problems," **Biological Cybernetics** 52, 141-152, 1985.

[Lin 90] Lin, T., H. Tseng, A Wu, N. Dogan and J. Meador, "Neural Net Diagnostics for VLSI Test," Proc. 2nd NASA SERC Symp. on VLSI Design, pp. 6.1.1-6.1.11, 1990.

[Meador 91] Meador, J., A. Wu, C.T. Tseng and T.S. Lin, "Fast Diagnosis of Integrated Circuit Faults Using Feedforward Neural Networks," IEEE Int. J. Conf. on Neural Networks, pp. I-269:273, 1991.

[Meador 93] Meador, J. and I. Pirvulescu, "A Stochastic Elastic Algorithm for the Traveling Salesman Problem," Proc. INNS World Congress on Neural Networks, pp. IV-428:431, 1993.

[Shen 92] Shen, T., Gan, J., and Yao, L., "A Generalized Placement Algorithm Based on Self-Organizaiton Neural Network," IEEE Int. J. Conf. on Neural Networks, pp. IV-761:766, 1992.

[Sriram 90] Sriram, M., and S.M. Kang, "A Modified Hopfield Network for Two-Dimensional Module Placement," IEEE Int. Symp. on Circuits and Systems, pp. 1664-1667, 1990.

[Wu 92] Wu, A. and J. Meador, "Data Driven Neural-Based Measurement Discrimination for Parametric Fault Diagnosis," 10th IEEE VLSI Test Symp., Atlantic City, NJ, April 1992.

[Wu 94] Wu, A., and J. Meador, "Measurement Selection for Parametric IC Fault Diagnosis," **J. Electronic Test Theory and Applications**, Vol 5 No. 1, Kluwer Academic, 1994 (to appear).

[Wu 94a] Wu, A. and J. Meador, "Simulation Sampling for Neural Based IC Parametric Fault Diagnosis," **J. Electronic Test Theory and Applications** (in review).

Neural Network for Isotope Identification

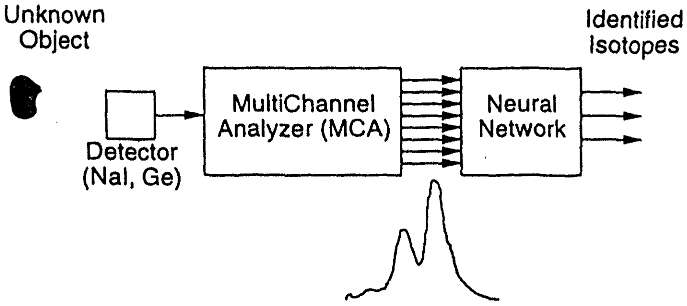
Paul E. Keller

Pacific Northwest Laboratory, Molecular Science Research Center, Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov

An optimal linear associative memory (OLAM) neural network that can be used to identify radioisotopes from their gamma-ray spectra is presented. The OLAM is useful in determining the composition of an unknown sample when the spectrum of the unknown is a linear superposition of known spectra. The neural network approach is useful in situations that require fast response but where precise quantification is less important. One feature of this technique is that is uses the whole spectrum in the identification process instead of individual peaks. For this reason, it is potentially more useful for processing data from lower resolution spectrometers. This approach has been tested with data generated by Monte Carlo simulations and with field data from both sodium iodide and germanium detectors.

Goals

- Examine the Applicability of Neural Networks in Isotope Identification (from a gamma spectrometer)
- Test a Neural Network with Simulated Data
- Test a Neural Network with Field Data

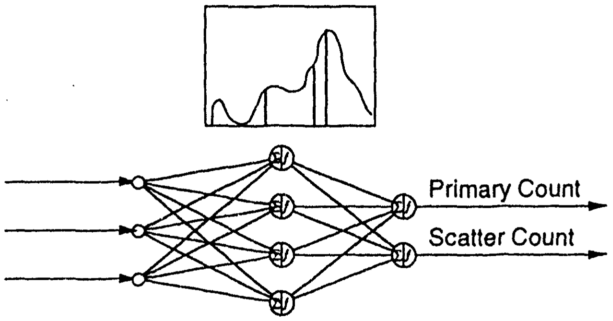


Approaches

Researchers	Institution	Neural Network
Koohi-Fayegh, Green, et. al.	University of Birmingham (UK)	Perceptron with Linear Outputs
Olmos, Diaz, Perez, et. al.	CIEMAT (Spain)	Optimal Linear Associative Memory
Ogawa, Niskizaki	Hosei University (Japan)	Backpropagation*

1. Classify Primary Counts and Scattered Counts

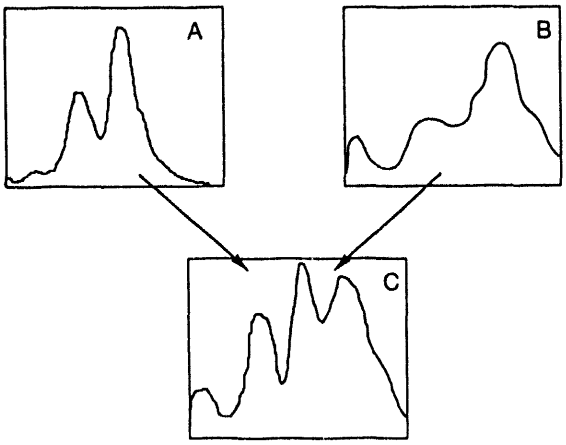
- Examine a few narrow windows in the spectrum
- Uses Feedforward Network Trained by Backpropagation



2. Gamma-Ray Spectroscopy

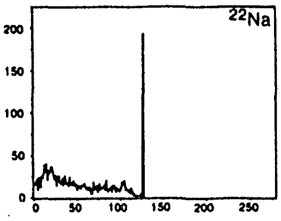
Purpose:	Determination of Isotope Composition of a Sample From its Gamma Ray Spectrum
Neural Network:	Optimal Linear Associative Memory
Configuration:	Single Layer Feed-Forward Network with Linear Nodes
Training Set:	Gamma-Ray Spectra
Inputs:	1 for Each Channel from Gamma Ray Spectrum
Outputs:	1 for Each Stored Isotope Spectra

Spectrum is Linear Superposition of Spectra

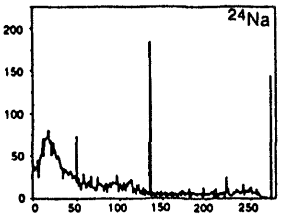


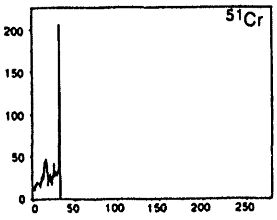
Optimal Linear Associative Memory

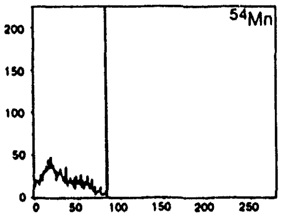
Training Patterns

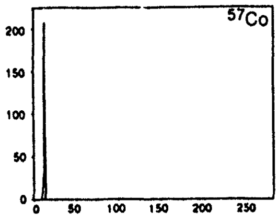


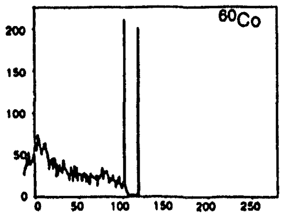
The training set consists of gamma ray spectra from 7 isotopes. These spectra were generated using a Monte Carlo simulation for a point source placed 10 cm from a Germanium detector.

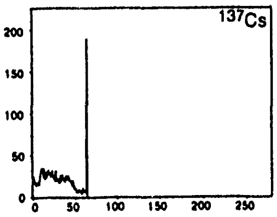




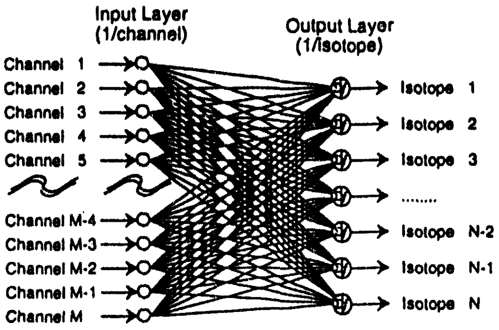






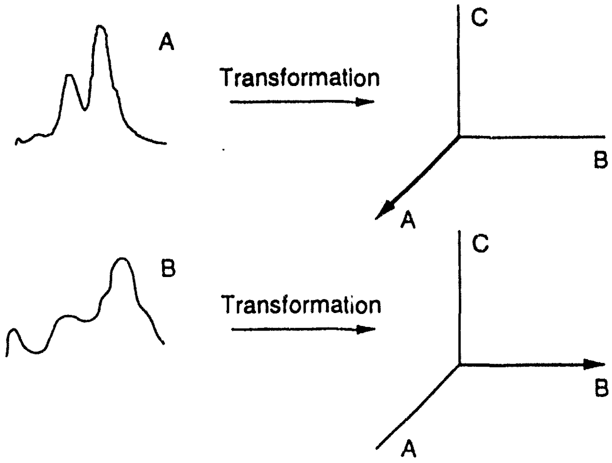


Neural Network Layout

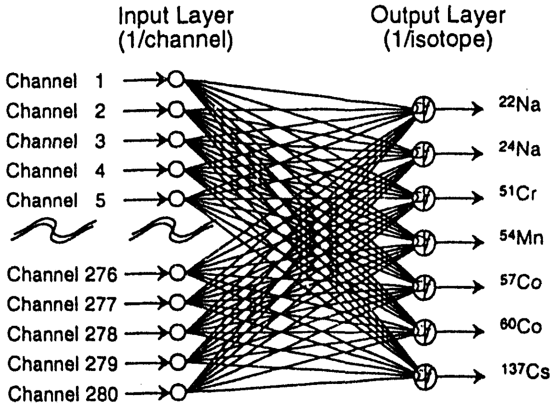


Number of Inputs = Number of Channels

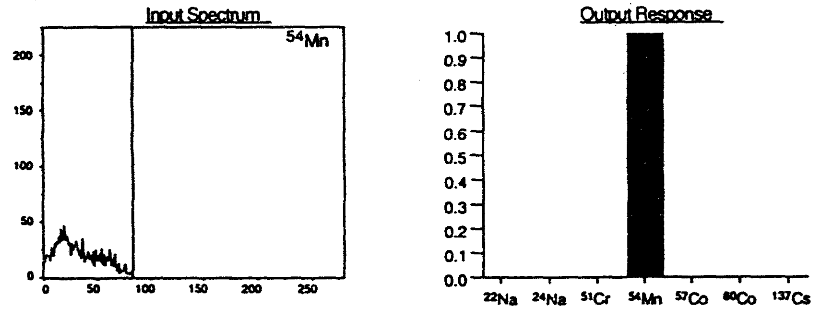
Number of Outputs = Number of Isotopes



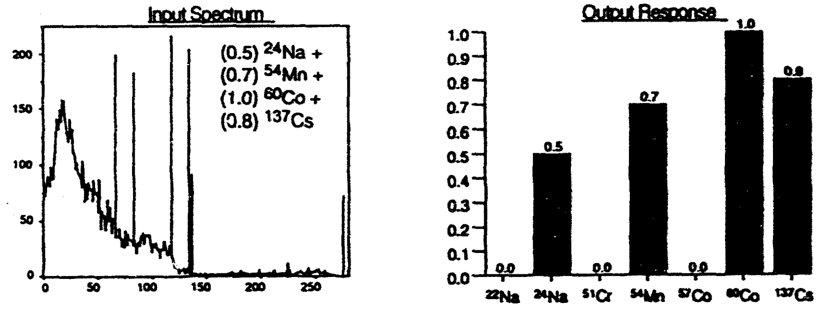
Neural Network Layout



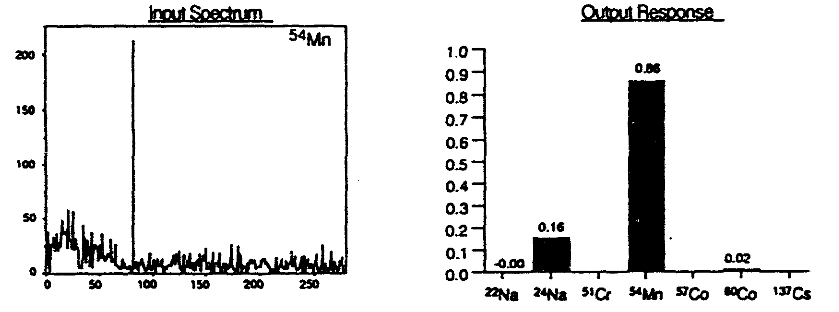
1. Perfect Recall of a Stored Pattern:



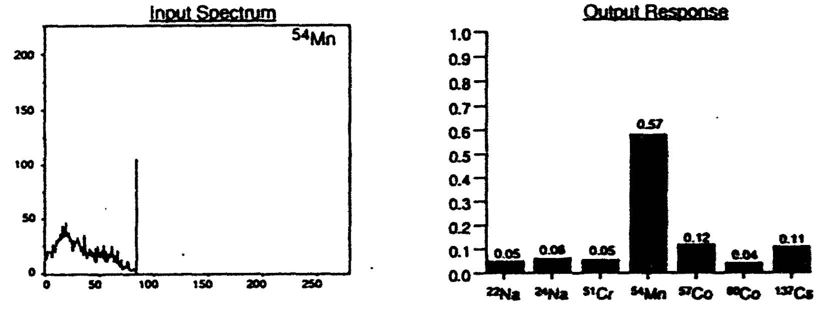
2. Identification of Sample Composition:



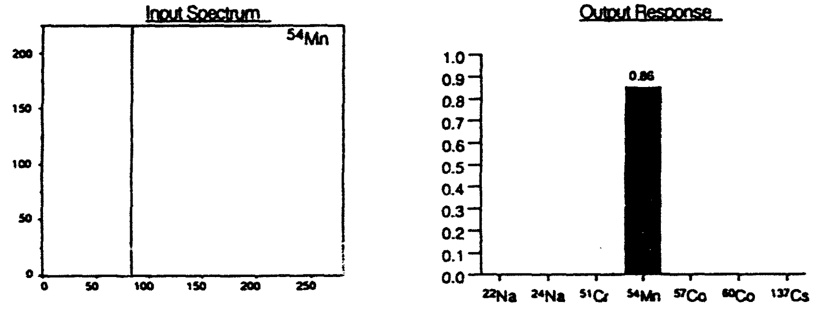
3. Recall from Spectrum with Additive Noise (σ = 5% of peak):



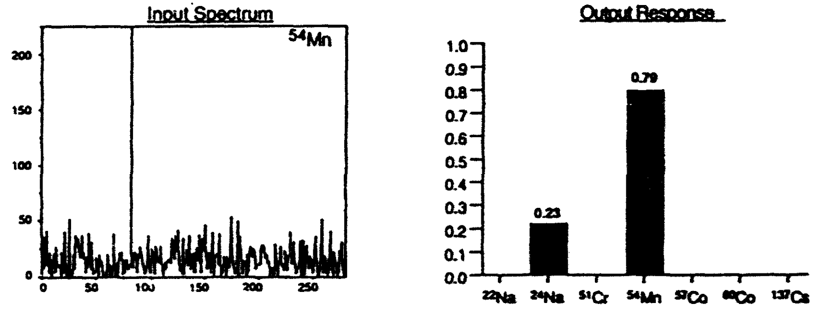
4. Recall from Spectrum with Peak Reduced to 50%:



5. Recall from Peak Alone:

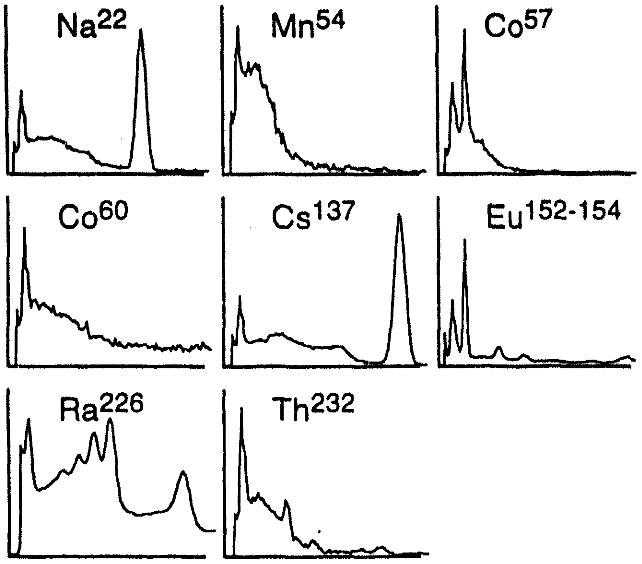


6. Recall from Peak and Additive Noise (σ=10% of peak):



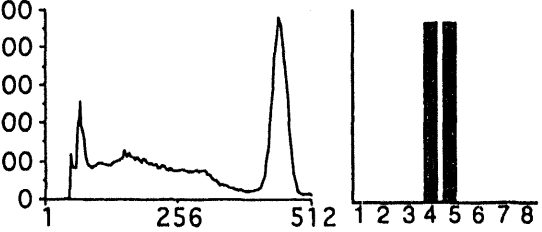
Training Patterns

The training set consists of gamma ray spectra from 8 isotopes. These spectra were collected in the field with calibration sources placed 1 meter from a Sodium Iodide detector. Each spectrum is composed of 512 channels of data.



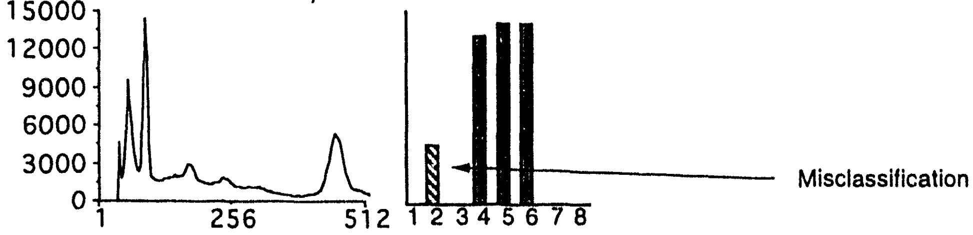
Spectrum ANN Output

Mixture of $Cobalt^{60}$ and $Cesium^{137}$



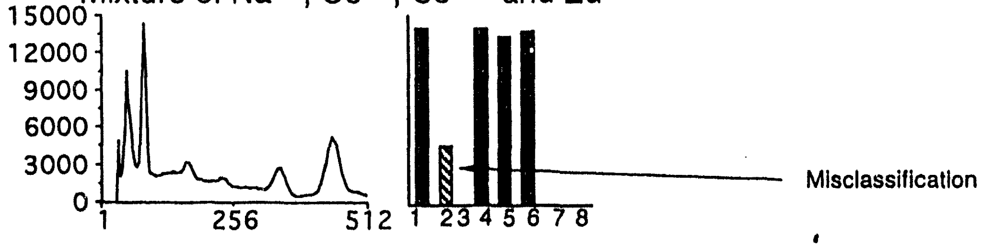
Spectrum ANN Output

Mixture of Co^{60}, Cs^{137} and $Eu^{152-154}$

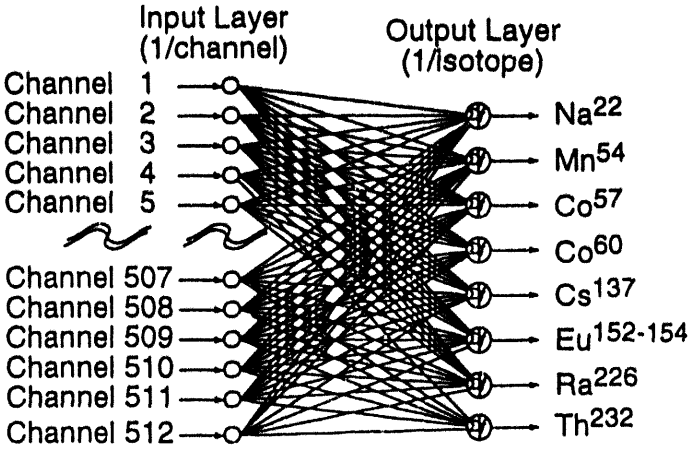


Spectrum ANN Output

Mixture of Na^{22}, Co^{60}, Cs^{137} and $Eu^{152-154}$



Neural Network Layout



Remarks

- Simple
  Training Consists of a Matrix Orthogonalization Process
- Fast
  4096 channel input with 10 outputs can be down in 100-150 mS on an Intel 486 DX at 33 MHz.
- Uses Whole Spectrum
- Potentially Useful with Low Resolution Spectrometers (e.g., NaI)

Areas for Further Work

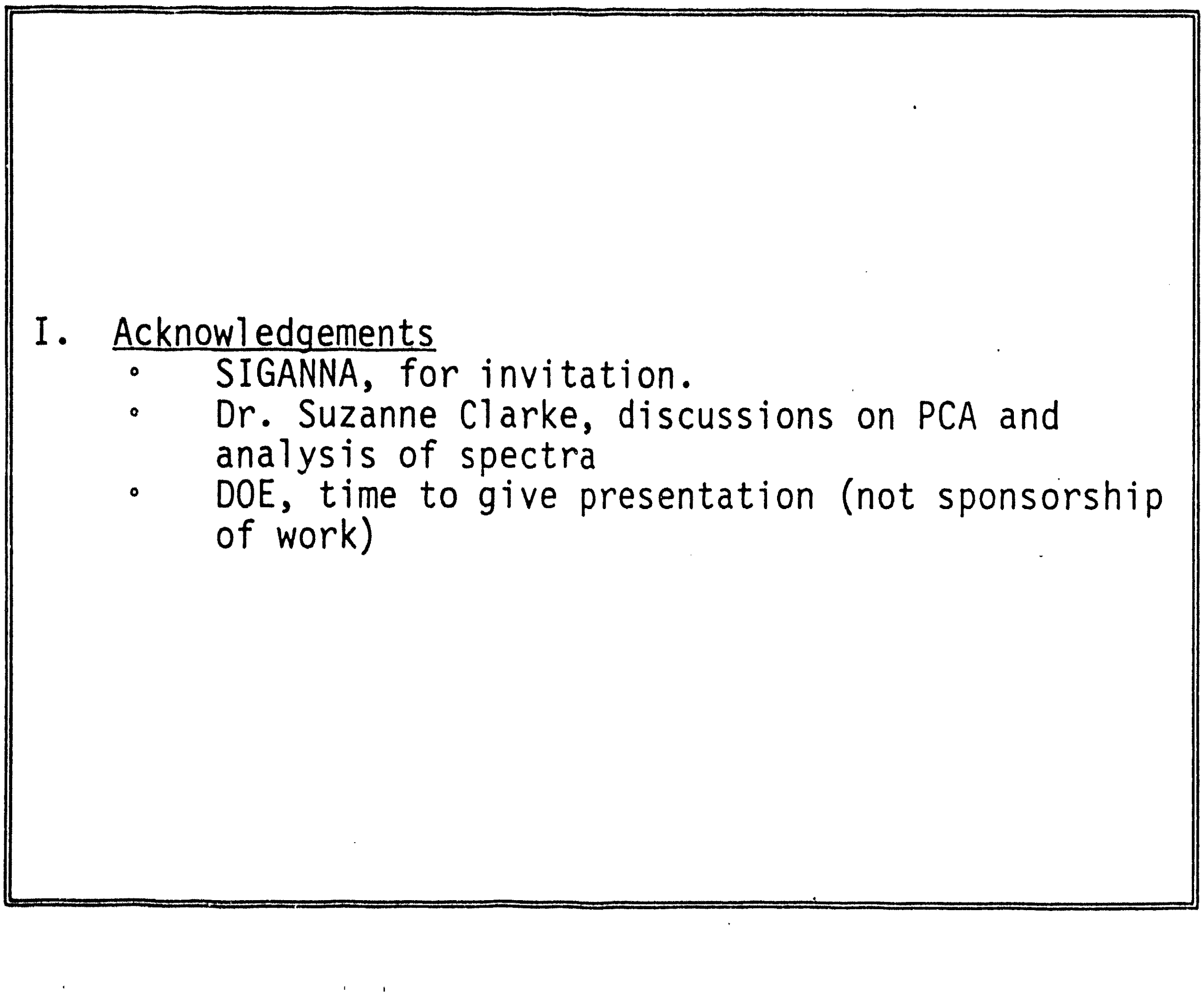
- Compensation for Gain Shifting
- Allow a Variety of Source Geometries
  (e.g., point source, broad source)

A Multicomponent Spectrum Decomposition Network

Harry Bell

U.S. Department of Energy, Richland Office
K6-05, P.O. Box 550, Richland, WA 99352
phone: (509) 376-9623

An algorithm and associated perceptron-like network are presented for the spectral decomposition of a multicomponent spectrum given a series of data vectors. Incorporated is an ad hoc algorithm of Edmund Malinowski's for the determination of the likely number of underlying additive components in a data matrix. The algorithm uses principal components analysis to determine the eigenvalues and associated eigenvectors. The Malinowski routine selects the minimal orthogonal basis set to reconstruct noise-free data vectors. At this point, any linear combination of this orthogonal basis set that results in "realistic" spectra can be explored. There are numerous "realistic" constraints that could then be used in further narrowing the possibilities for an optimal linear combination of the initial basis vectors, among these are: smoothness, unbiased deviations about a reconstructed data vector (maximizing the number of deviation sign changes), and positivity of the signal (if negative data is physically unrealistic). A genetic algorithm is used to fully explore the space of linear combinations to arrive at a most likely realistic basis set. Although a neural network is not necessary for a solution to this problem it aids in visualization of the problem and in suggesting further enhancements to the algorithm.



I. Acknowledgements

- SIGANNA, for invitation.
- Dr. Suzanne Clarke, discussions on PCA and analysis of spectra
- DOE, time to give presentation (not sponsorship of work)

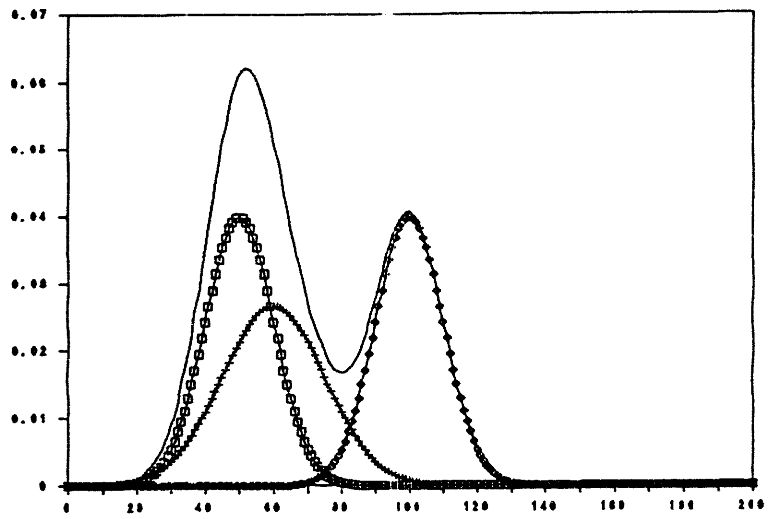


Figure 9 - Example of a composite spectrum - a sum of three gaussian curves

II. <u>Problem statement</u>

- For each of K composite measured spectra measured at J energy levels per spectrum, (data matrix s_{jk} for all j = 1 to J and k = 1 to K) find number of significant components, I, their concentrations, c_{ik} and absorptivities, w_{ij}

$$s_{j_k} = \sum_{i=1}^{I} c_{i_k} \cdot w_{i_j}$$

- Applies to linear chemical spectroscopies (eg. UV-VIS) also may be applicable to other kinds of analysis.
- Potential applications for mixture analysis with uncalibrated and/or ephemeral solution species.

III. <u>Step 1 - How many real components in the mixture given by s_{jk} (measured spectra)?</u>

- Much progress in this area in the last few decades
- Equals the number of significant eigenvalues of the covariance matrix:

$$cov_{i_j} = \sum_{k=1}^{K} s_{i_k} \cdot s_{j_k}$$

- Effective algorithms are readily available
  1. Barry Wise PNL (earlier presentation at this workshop)
  2. Edwin Malinowski, <u>Factor Analysis in Chemistry, 2nd ed.</u> contains a Matlab implementation.

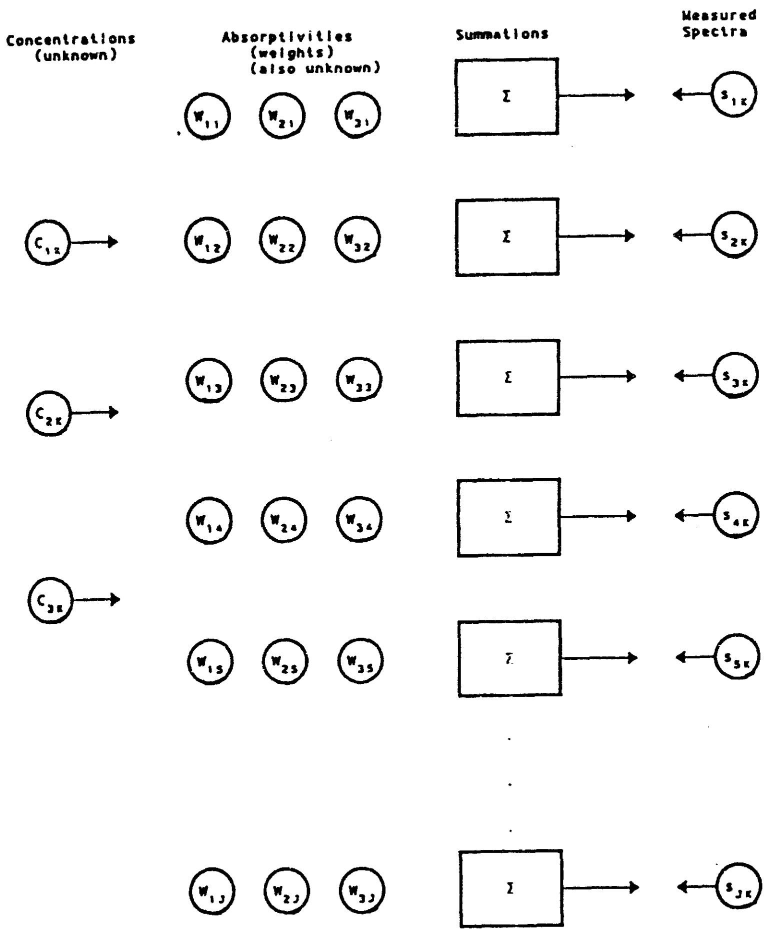


Figure 8 - Basic network structure - a perceptron without the squashing function with unknown inputs, C_{ik}, (connections are not drawn).

V. Straight perceptron approach

- Both individual c_{ik} and w_{ij} treated as independent unknowns
- Reduced to a simple, well-defined (but vast) optimization problem
- May need to reduce the search space since too many unknowns
- Solve for both concentrations and absorptivities using gradient descent methods

VI. Alternative - direct manipulation of eigenvectors of covariance matrix

- Principal eigenvectors of the spectra covariance matrix (COV_{ij}) form a basis for the absorptivities, w_{ij}.
- Need to find a linear combination of these basis vectors minimizing the following:
  1. Resulting absorptivities that are negative
  2. Resulting concentrations that are negative
  3. Highly correlated absorptivities
- Still a high dimensional optimization (9 dimensions for a 3-component system).

VII. Iterative procedure - direct eigenvector manipulation

1. select linear combination of eigenvectors, candidate absorptivities.
2. generate concentrations based on these absorptivities using linear regression
3. determine various metrics on the absorptivities and concentrations such as:
   a. extent of correlation between the candidate absorptivities
   b. positivity of absorptivities
   c. positivity of concentrations
4. evaluate an objective function using above performance metrics
5. repeat starting at step 1 if objective function is unsatisfactory.

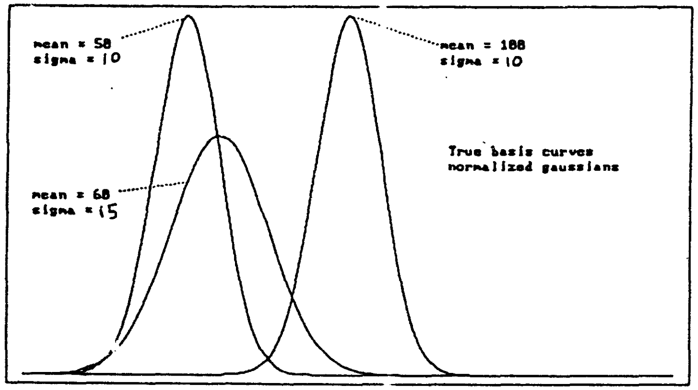


Figure 5 - True underlying absorptivities of spectra

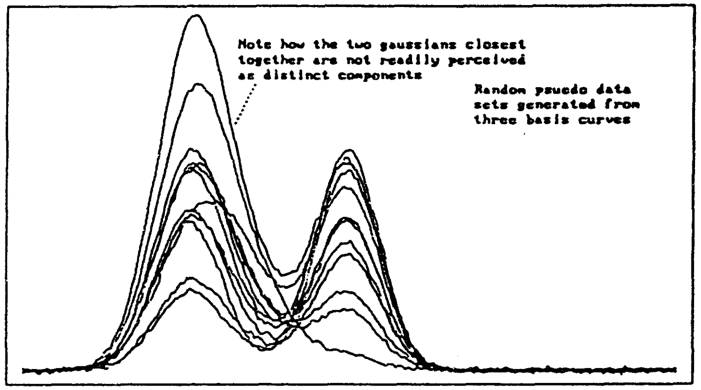


Figure 6 - Random spectra derived from true absorptivities

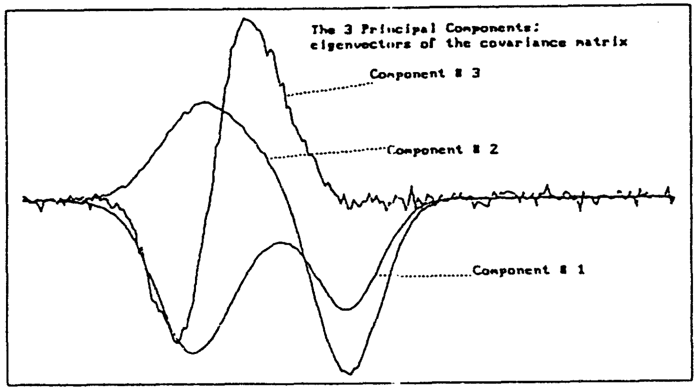


Figure 7 - Unaltered eigenvectors from principal components analysis

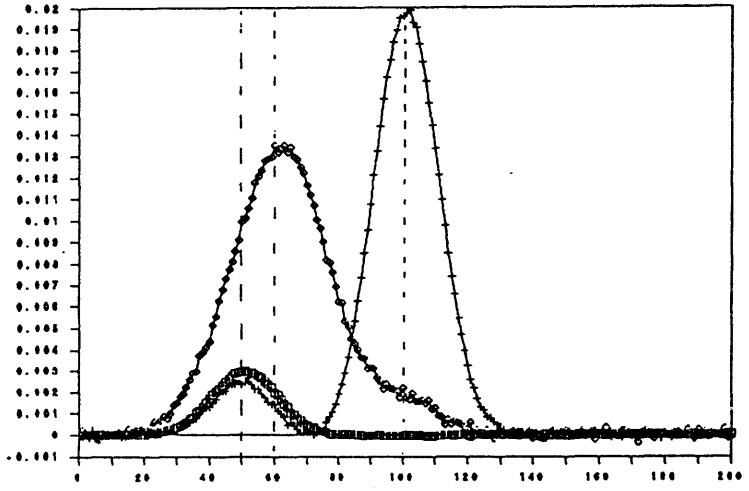


Figure 4 - Results of direct eigenvector optimization

Neural Network Based Chemical Sensor Systems

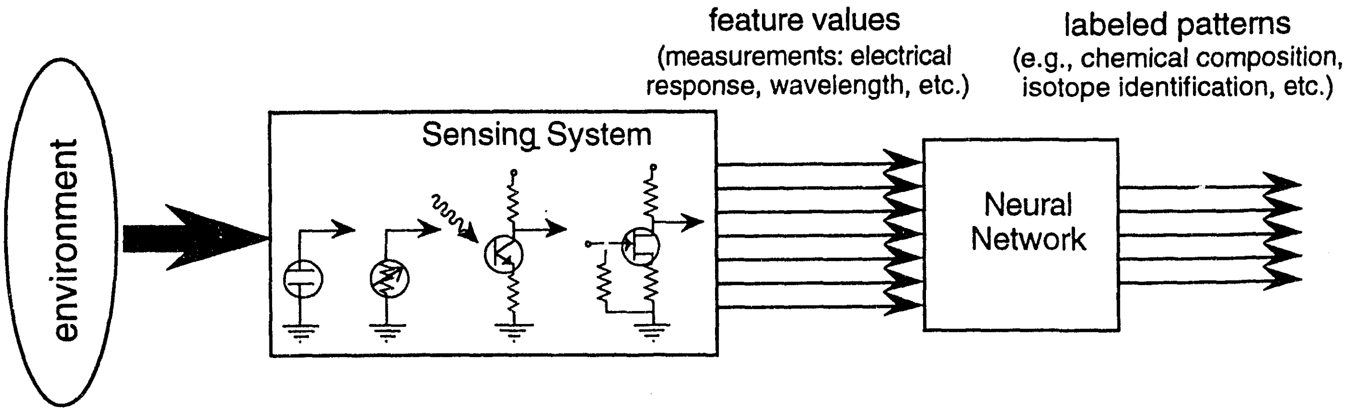
Paul E. Keller

Pacific Northwest Laboratory, Molecular Science Research Center,
Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov

Compact, portable, and inexpensive systems capable of quickly identifying contaminants in the field are of great importance when monitoring the environment. One approach is to combine a sensor array with a neural network. One advantage of this approach is that most of the intense computation takes place during the training process. Once the neural network is trained for a particular task, operation consists of propagating the data through the neural network. In this presentation, two prototype chemical-sensing systems are discussed. These prototypes are currently being used in an evaluation of the application of neural networks to chemical sensor analysis.

One prototype consists of an array of tin-oxide gas sensors and can be used to identify chemical vapors. Although each sensor is tuned to a specific chemical vapor, each responds to a wide variety of chemicals. Collectively, these sensors respond with unique signatures (patterns) to chemical vapors. During the training process, known mixtures of various chemical vapors are presented to the system. The responses of all the sensors provide a set of training patterns for the neural network. During operation, the system can rapidly identify the composition of a vapor provided that the vapor is composed of chemicals used during the training process.

The other prototype consists of an array of optical sensors that can be used to identify liquids by their absorption spectra. Light is passed through the liquid and into the sensor array. By examining the absorption at different wavelengths, the neural network is able to identify the chemicals in the liquid. Each optical sensor is composed of a silicon detector covered by a narrow bandpass (10 nm) interference filter and is sensitive to a specific wavelength of light.

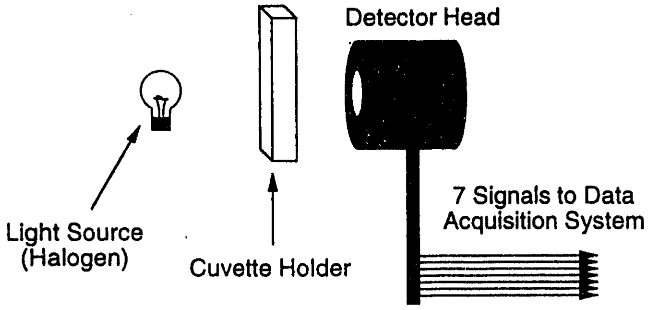

feature values
(measurements: electrical response, wavelength, etc.)
labeled patterns
(e.g., chemical composition, isotope identification, etc.)
environment
Sensing System
Neural Network

Neural Network Based Artificial Noses

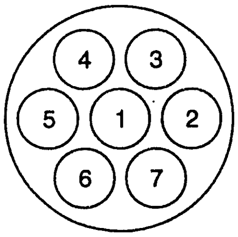
Application	Sensor Array	Neural Network	Institution
Identification of Whiskey by Odor	8 Quartz Resonators	• Fuzzy LVQ • LVQ • Backpropagation	Tokyo Institute of Technology, Japan
Gas Identification (NH_3, Acetone, Hexane)	3 Tin Oxide Sensors	• Backpropagation	Tokyo Institute of Technology, Japan
Identification of Ions in Blood (Na^+, K^+, etc.)	PVC Membrane Sensors	• Backpropagation	Dublin City University, Ireland
Identification of Ions (Na^+, K^+, Ca^+, Mg^+)	7 Ion Selective Electrodes (ISE)	• Counterpropagation • Backpropagation • OLAM & Backprop.*	
Food and Beverage Odor, Perfume, and Alcohol Identification	12 Tin Oxide Sensors	• Backpropagation	University of Warwick, England
Gas Identification (H_2, NH_3, Ethanol, Ethylene)	6 Metal Oxide Field Effect Transistors (MOSFET)	• Backpropagation	Linköping Institute of Technology, Sweden
Analysis of Motor and Turbine Fuels	10 Tin Oxide Sensors	• Hopfield, • Hamming • Boltzmann	Oak Ridge National Lab, USA

LVQ = Learning Vector Quantization OLAM = Optimal Linear Associative Memory
*Orthogonalized input equivalent to the Optimal Linear Associative Memory (OLAM)

Optical Sensor Array

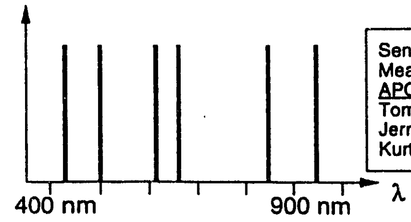


1. Unfiltered
2. 415 nm (10 nm bandpass)
3. 500 nm (10 nm bandpass)
4. 610 nm (10 nm bandpass)
5. 665 nm (10 nm bandpass)
6. 842 nm (10 nm bandpass)
7. 940 nm (10 nm bandpass)



Sensor Layout

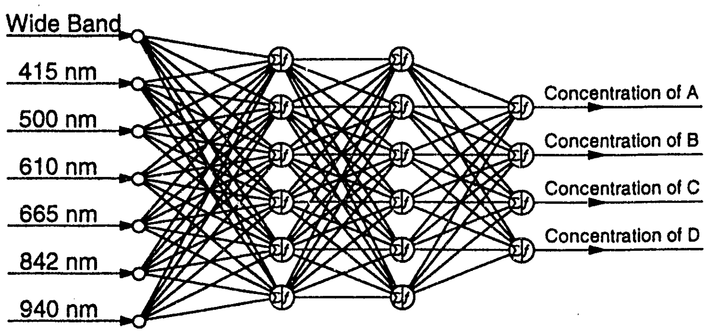
Sampling Response



Sensor Physics Group, Automation and Measurement Sciences Department
APC, PNL
Tom Sloane
Jerry Berndt
Kurt Stahl

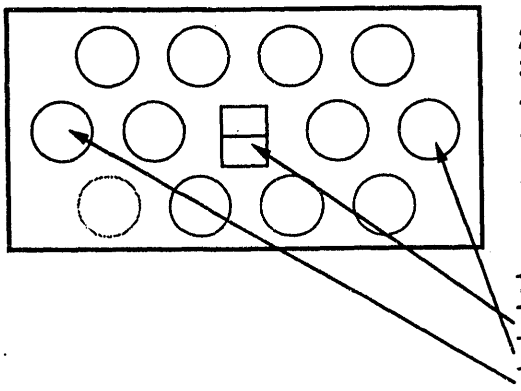
Determine Concentration of Different Dyes

- Measure Absorption at 6 Wavelengths
- Feed Response Values at These 6 Wavelengths into Neural Network

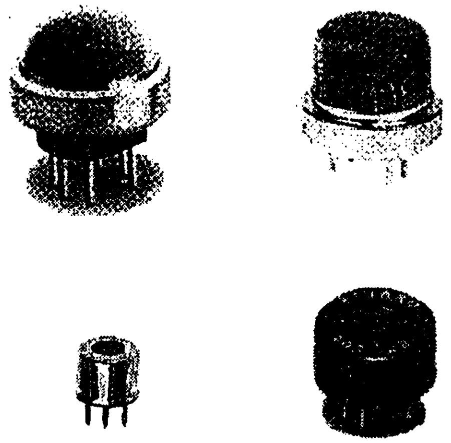


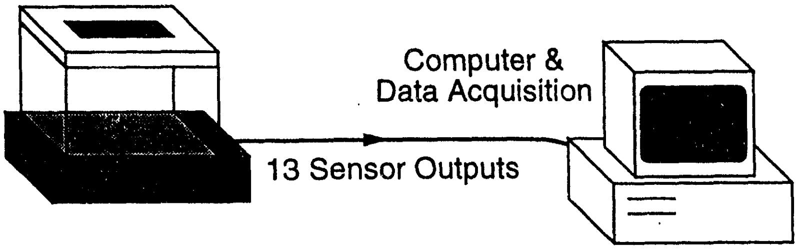
(Values Range From 0 to 5 Volts)

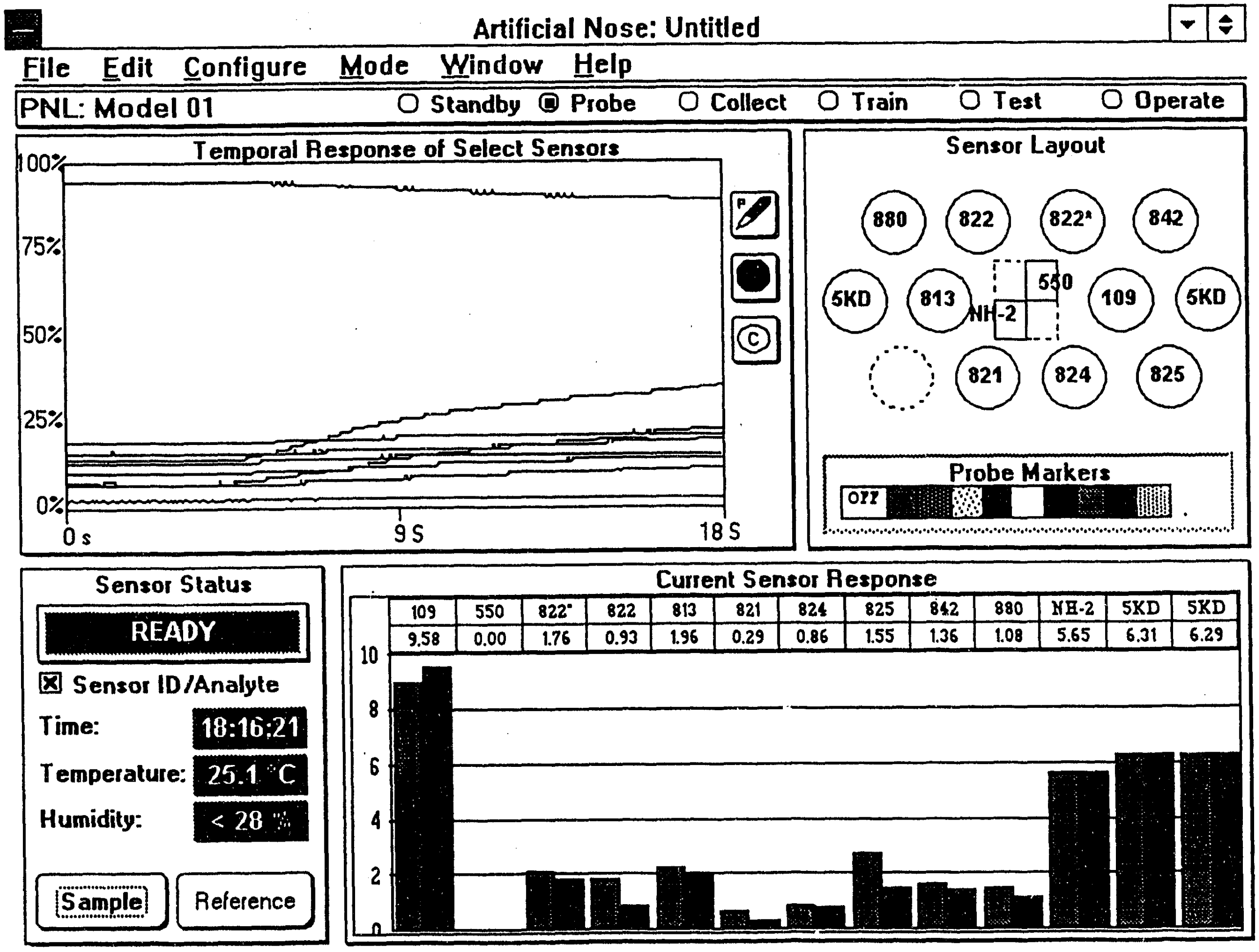
Chemical Sensor Array



13 Sensors
1. TGS 109
2. TGS 550
3. TGS 813
4. TGS 821
5. TGS 822
6. TGS 822
7. TGS 824
8. TGS 825
9. TGS 842
10. TGS 880
11. NH-2 Humdity Sensor
12. Thermistor
13. Thermistor







Artificial Nose: Untitled

File Edit Configure Mode Window Help

PNL: Model 01 ○ Standby ● Probe ○ Collect ○ Train ○ Test ○ Operate

Current Sensor Response

109	550	822*	822	813	821	824	825	842	880	NH-2	5KD	5KD
9.58	0.00	1.76	0.93	1.96	0.29	0.86	1.55	1.36	1.08	5.65	6.31	6.29

Sensor Status

READY

☒ Sensor ID/Analyte

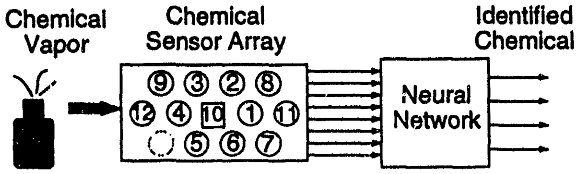
Time: 18:16:21

Temperature: 25.1 °C

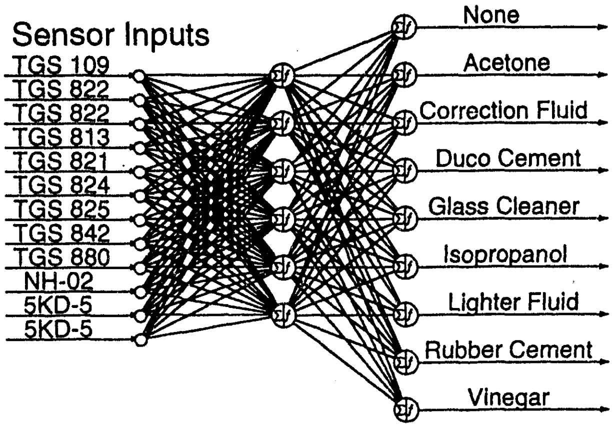
Humidity: < 28 %

Sample Reference

System



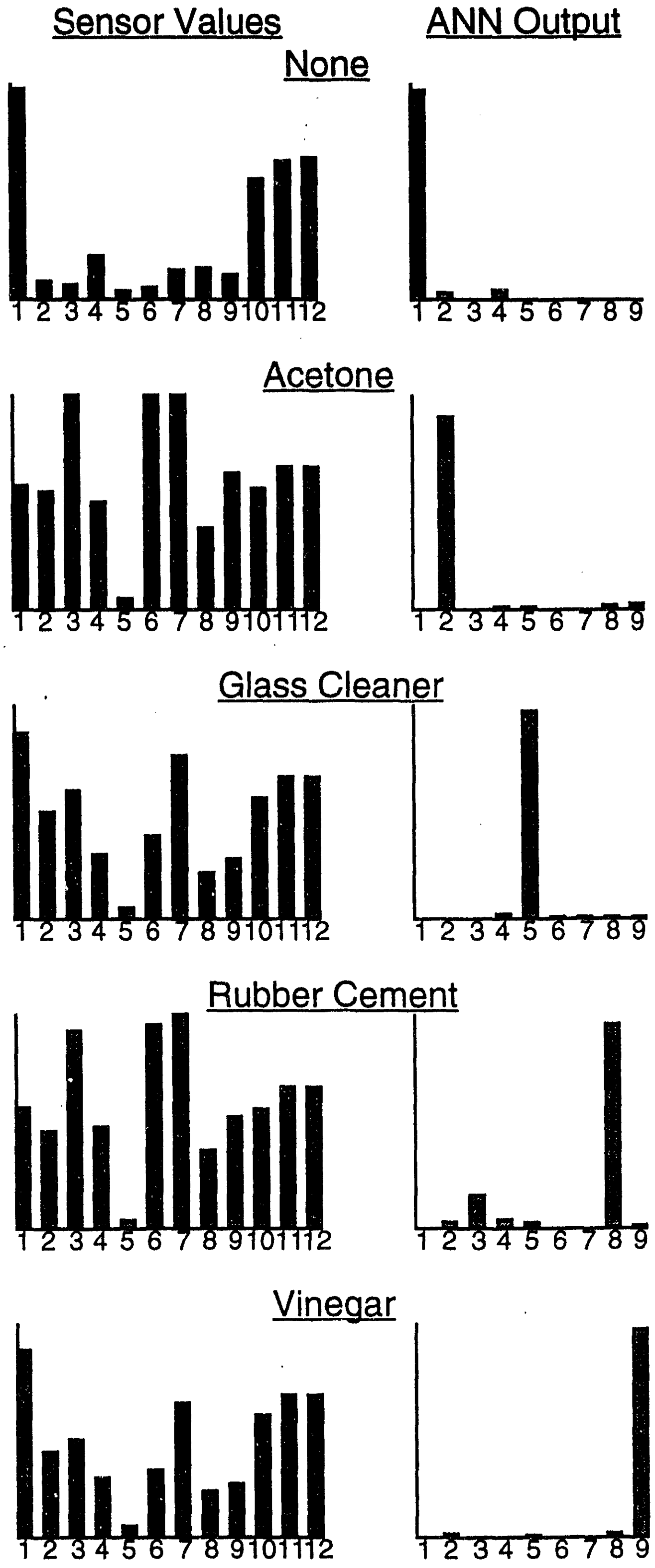
Neural Network Layout



Training Parameters

Type:	Backpropagation in Batch Mode
Architecutre:	12-6-9
Activation:	Logistic
Learning Rate:	0.01
Momentum:	0.9
No. of Epochs:	15000

Results



Real-Time Data Analysis via Artificial Neural Network

- Fast Classification: Most time is spent training the neural network
- Operation:

	Processing Rate	Small System	Medium System	Large System
Software Implementations:		(16-16-16)	(256-256-256)	(4096-4096-4096)
Intel 80386/16MHz	$4.4 \bullet 10^{4}$ CPS	15 mS	3.0 S	13 min*
Intel 80486/33MHz	$1.1 \bullet 10^{5}$ CPS	6 mS	1.2 S	5.1 min *
VAX 8600	$2.0 \bullet 10^{5}$ CPS	7 mS	68.0 mS	2.8 min *
Sun SPARCstation 10	$1.3 \bullet 10^{6}$ CPS	440 µS	100 mS	26 S *(69)
DECstation 5000	$1.3 \bullet 10^{6}$ CPS	570 µS	100 mS	26 S *(45)
Electronic Hardware Implementations:				
ExploreNet 3000 w/SNAP-64	$1.0 \bullet 10^{9}$ CPS		~ 130 µS	~35 mS
Adaptive Solutions CNAPS	$<5.0 \bullet 10^{9}$ CPS	-	~26 uS	<7 mS
Intel 80187	$<1.6 \bullet 10^{10}$ CPS	-	~8 uS	<2 mS
Optical Hardware Implementations:				
Planar Holograms & Light Modulators	$2.2 \bullet 10^{11}$ CPS	-	-	200 uS
Planar Holograms & Laser Diodes	$4.5 \bullet 10^{13}$ CPS	-	-	1 µS

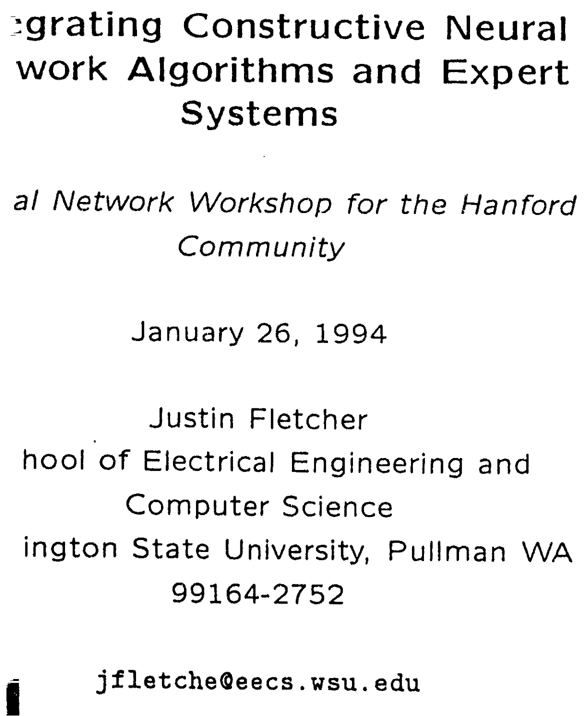
CPS: Connections per Second
* Projected processing time assuming more than 128 MBytes of free memory. Actual processing times, where available, are noted in parenthesis.

Integrating A Parallel Constructive Neural Network Algorithm with an Expert System

Justin Fletcher and Zoran Obradoviĉ

Washington State University, School of Electrical Engineering and Computer Science
Pullman, WA 99164-2752
phone: (509) 335-2217 or (509)838-0164 fax: (509) 335-3818
internet: jfletche@eecs.wsu.edu

An improved parallel constructive neural network learning algorithm is presented which generates a near-minimal architecture. Traditional neural network learning involves modification of the interconnection weights between neurons on a pre-specified network. Determining that network architecture is a challenging problem which currently requires an expensive trial-and-error process. Rather than learning on a pre-specified network topology, a constructive algorithm also learns the topology in a manner specific to the problem. Experimental results are presented that indicate that a near-minimal architecture is created with improved generalization. Secondly, the integration of expert systems with constructive neural network algorithms is examined. The motivation is to combine pre-existing knowledge about the problem domain with information derived from examples. The concepts of knowledge-based systems and machine learning are combined by integrating an expert system with the constructive neural network learning algorithm. Two approaches are explored: embedding the expert system directly and converting the expert system rule base into a neural network. This initial system is then extended by constructively learning additional hidden units in a problem-specific manner. Results indicate that combining pre-existing knowledge with knowledge gained from examples may result in improved prediction quality.



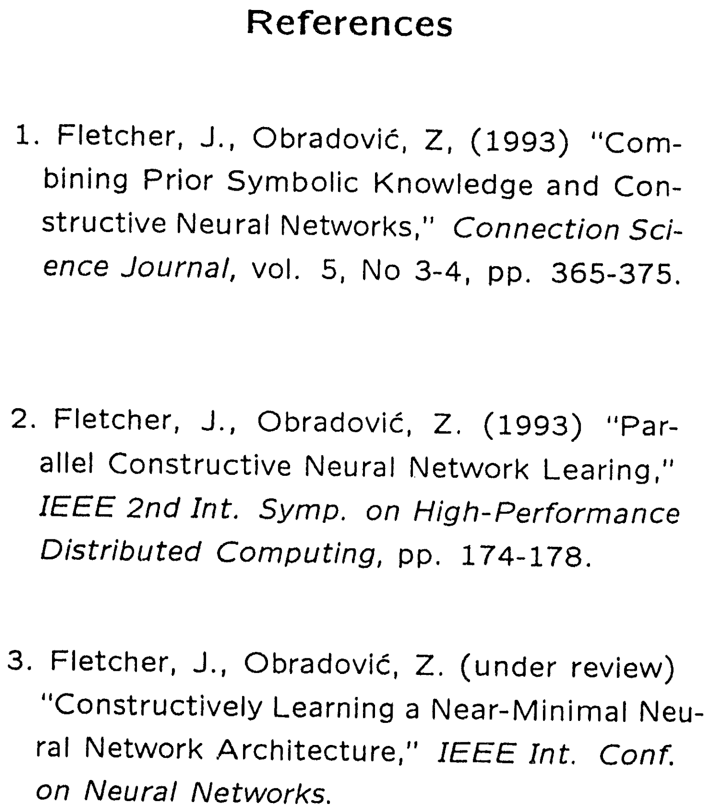
grating Constructive Neural
work Algorithms and Expert
Systems

*al Network Workshop for the Hanford
Community*

January 26, 1994

Justin Fletcher
hool of Electrical Engineering and
Computer Science
ington State University, Pullman WA
99164-2752

jfletche@eecs.wsu.edu



References

1. Fletcher, J., Obradović, Z, (1993) "Combining Prior Symbolic Knowledge and Constructive Neural Networks," *Connection Science Journal*, vol. 5, No 3-4, pp. 365-375.

2. Fletcher, J., Obradović, Z. (1993) "Parallel Constructive Neural Network Learing," *IEEE 2nd Int. Symp. on High-Performance Distributed Computing*, pp. 174-178.

3. Fletcher, J., Obradović, Z. (under review) "Constructively Learning a Near-Minimal Neural Network Architecture," *IEEE Int. Conf. on Neural Networks*.

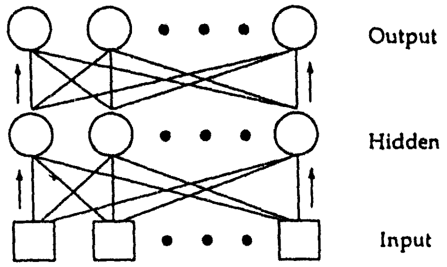
Contents

- Introduction to Neural Networks
- Introduction to Hybrid Systems
- Introduction to Constructive Neural Network Algorithms
- Hyperplane Determination from Examples Alone
- Integration with Expert Systems

Introduction to Neural Networks

Neural Network: weighted graph

Feed-forward NN: weighted directed graph



Weighted sum: $\sum_{i=1}^{n} w_i x_i$

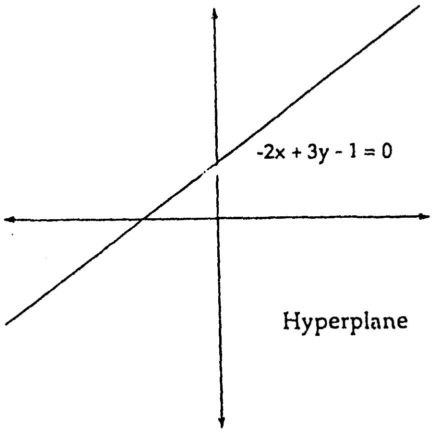
Binary neural network

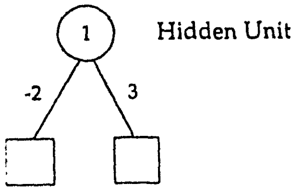
Activation function $g(x)$: R → $\{0,1\}$,

$$g(x) = \begin{cases} 0 & \text{if } x < t \\ 1 & \text{if } x \geq t \end{cases}$$

for some $t \in$ R.

Relationship between Hyperplanes and Hidden Units





Introduction to Hybrid Systems

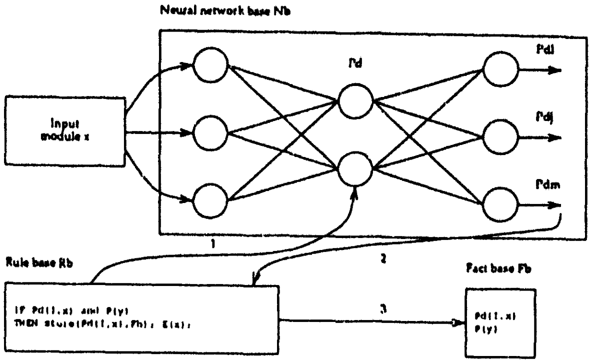
Rationale

- An expert system contains pre-existing knowledge.
- The knowledge may be incomplete or contradictory.
- By combining the expert system with a machine learning approach (a constructive neural network algorithm), the overall performance of the system will be improved.

Techniques for Integration

- Embed a neural network into an expert system
- Convert the expert system to a neural network
- Integrate the results of multiple classifiers
- Integrate the expert system without conversion

Embed a Neural Network into an Expert System



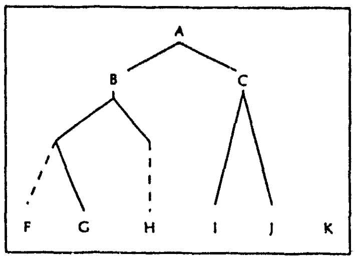
Convert the Expert System to a Neural Network

A: B, C

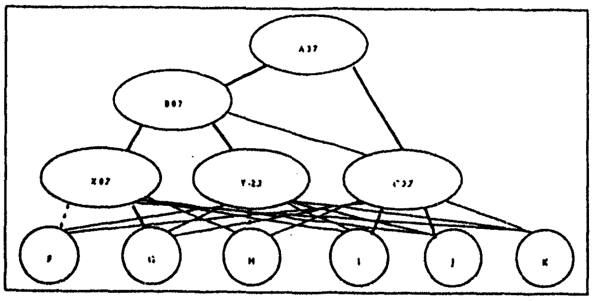
B: not F, G
B: not H

C: I, J

Rule Base

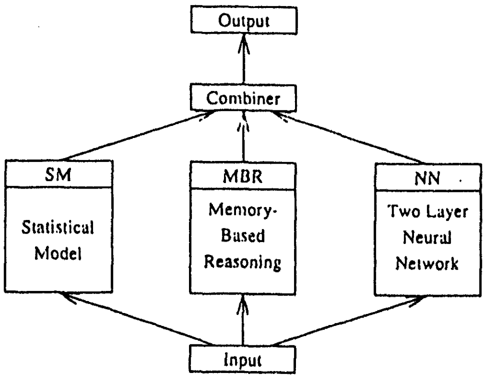


And / Or Graph



Neural Network

Integrate the Results of Multiple Classifiers



Introduction to Constructive Neural Network Algorithms

Topology Selection Objectives

- Large enough to define separating surface
- Small enough to generalize well

Traditional NN Learning

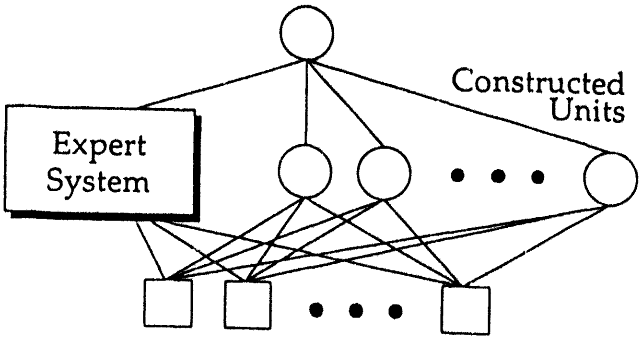
- Pre-specified architecture
- Supervised learning
- Modification of connection weights
- Training and test sets

Constructive NN Learning

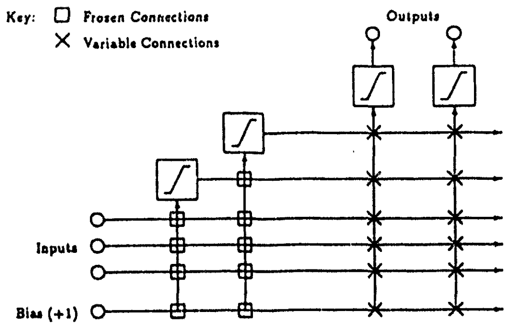
- Architecture learned to fit problem

Integrate the Expert System Without Conversion

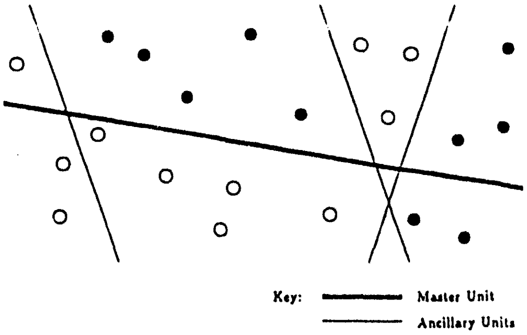
Integrated System



Cascade-Correlation

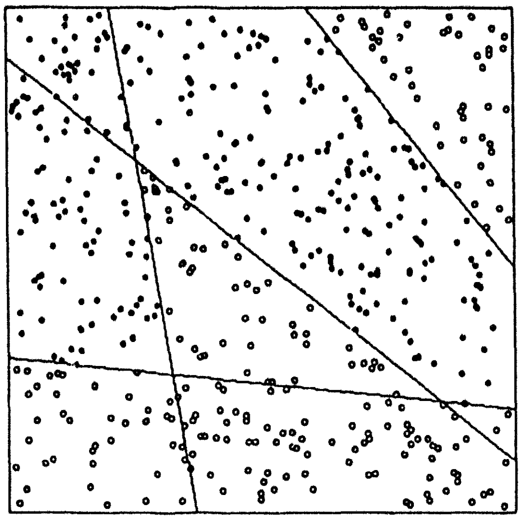


Tiling



Hyperplane Determination from Examples Alone

Determine separating hyperplanes as close to the optimal separating hyperplanes as possible.

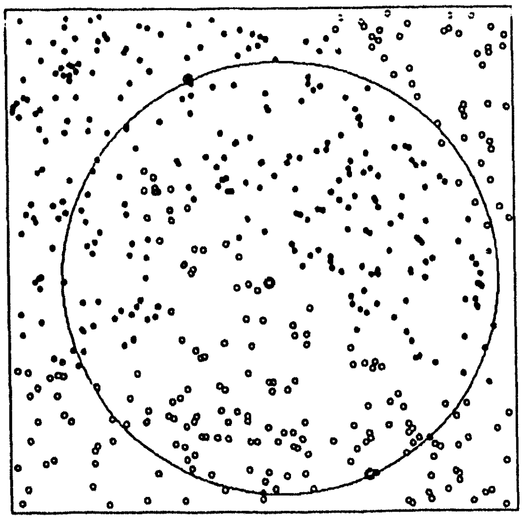


Steps:

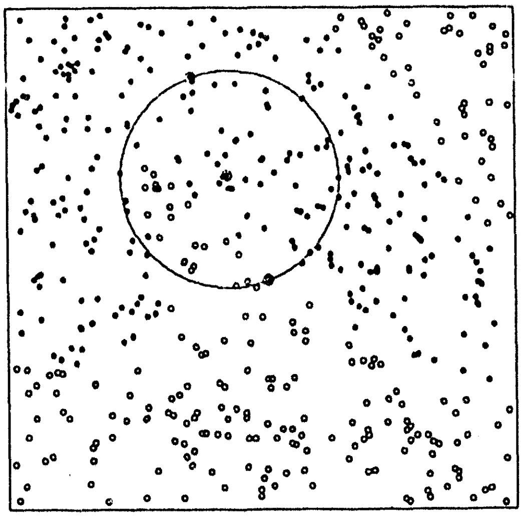
1. Determination of points on separating hyperplanes
2. Determination of candidate hyperplanes from separating points
3. Creation of hidden units from selected hyperplanes

Determination of Points on Separating Hyperplanes

From examples of opposing classes, select the closest example to the midpoint.

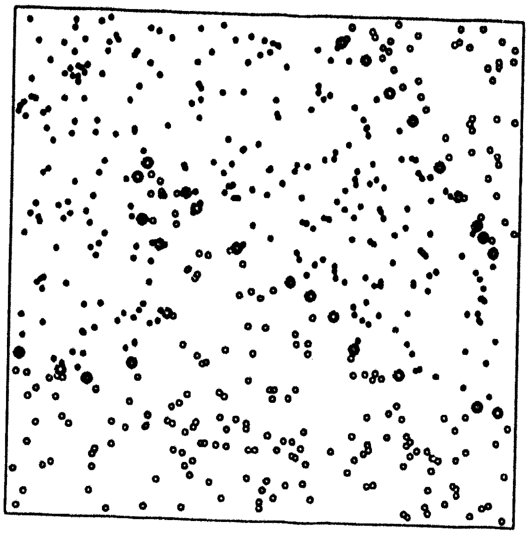


Repeat with the selected example and the endpoint of the opposite class.



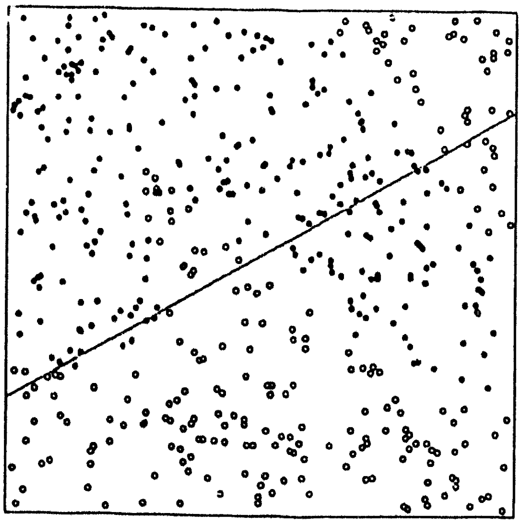
Continue until no example can be found. The midpoint is a point on the separating hyperplane. Repeat until:

- A pre-determined number of separating points have been found or
- A number of data points have been examined without finding a new separating point.



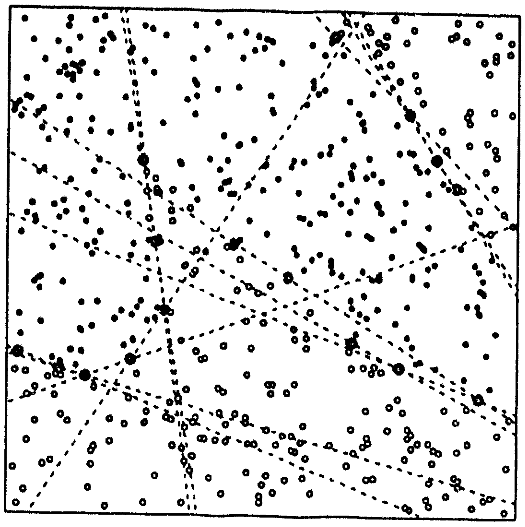
Creation of Hidden Units from Selected Hyperplanes

The first hidden unit is created from the candidate hyperplane which best classifies the training data.

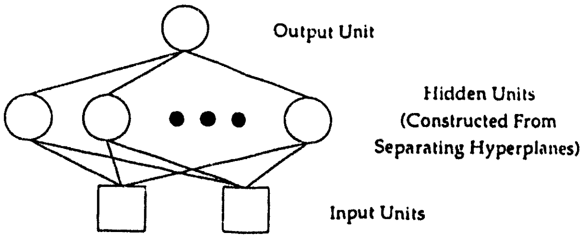


Determination of Candidate Hyperplanes from Separating Points

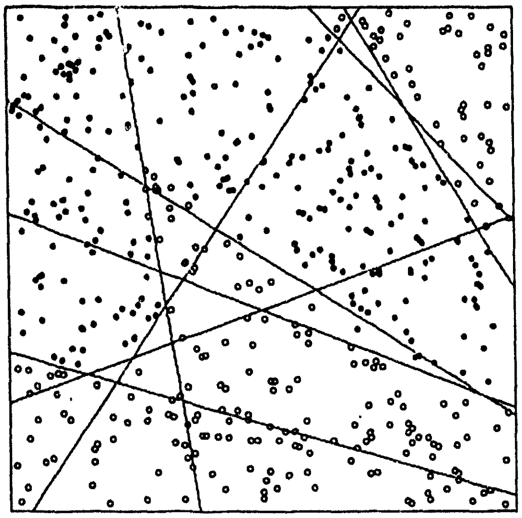
Select the nearest separating points to each separating point. Generate the separating hyperplane from the determined points.



A hidden unit is constructed from the separating hyperplane.



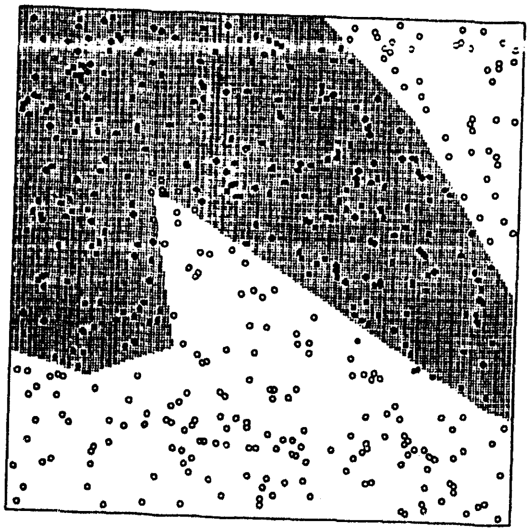
Remaining nyperpianes are createa by parallel evaluation of each of the candidate hyperplanes with the existing hidden units. This process is repeated until no significant classification improvement on the training data is obtained.



Output Layer Weights

The pocket algorithm is used to determine the output layer weights.

The resultant separating surface is determined by final training of the output layer weights.



MONK's Problems

- Binary classification problem
- Six features
- Problem 1: DNF
- Problem 2: Similar to parity
- Problem 3: DNF with 5% noise

Sequential Results

Percentage accuracy on the MONK's problems

Accuracy	Average # of Hidden Units	Average	
		Train	Test
Problem 1	4.8	99.19	96.66
Problem 2	6.3	73.60	66.73
Problem 3	5.7	96.14	94.00

Accuracy	Average # of Hidden Units	Best Case	
		Train	Test
Problem 1	4.8	100.00	100.0
Problem 2	6.3	78.11	68.98
Problem 3	5.7	98.36	97.22

Accuracy	Average # of Hidden Units	Worst Case	
		Train	Test
Problem 1	4.8	97.58	88.43
Problem 2	6.3	62.13	63.89
Problem 3	5.7	93.44	87.50

Parallel Hidden Unit Creation

- Master reads in data
- Master broadcasts data
- Master selects first hyperplane

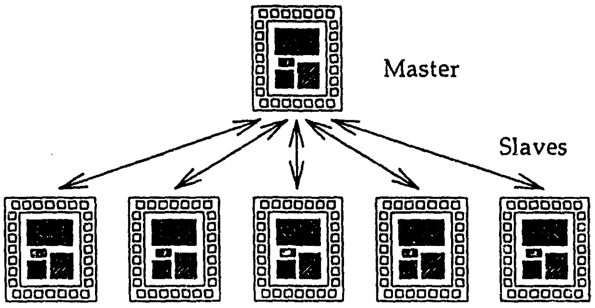
Repeat

- Master broadcasts hidden unit
- Slaves request candidate hyperplanes to evaluate
- Master selects best candidate hyperplane and creates hidden unit

while significant performance improvements

Parallel Hidden Layer Construction

Master / Slave Architecture



Implementation

The implementation was developed using *p4* in a distributed environment of DECStation 5000s, then ported to the CalTech Touchstone DELTA.

Parallel Results

Percentage accuracy on the MONK's problems

Sequential	Train	Test
Problem 1	100.00	85.42
Problem 2	81.07	70.37
Problem 3	94.67	72.69

Dist. Parallel	Train	Test
Problem 1	100.00	84.72
Problem 2	94.67	72.69
Problem 3	96.72	72.92

Touchstone	Train	Test
Problem 1	100.00	80.79
Problem 2	92.31	71.30
Problem 3	100.00	77.55

Addition of Hidden Units to Integrated System

If the expert system cannot correctly classify a given data point,

1. Determine a separating hyperplane.
2. Construct a hidden unit from the separating hyperplane.
3. Repeat as required.

Integration with Expert Systems

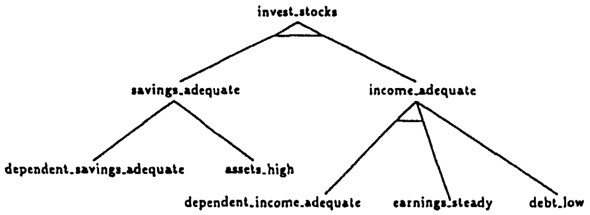
Expert System Conversion

Create an initial network from the rule base similar to Towell et. al.

Convert the rule base

if (savings_adequate and income_adequate)	then	invest_stocks
if dependent_savings_adequate	then	savings_adequate
if assets_high	then	savings_adequate
if (dependent_income_adequate and earnings_steady)	then	income_adequate
if debt_low	then	income_adequate
if (savings $\geq$ dependents $\times$ 5000)	then	dependent_savings_adequate
if (income $\geq$ 25000 + dependents $\times$ 4000)	then	dependent_income_adequate
if (assets $\geq$ income $\times$ 10)	then	assets_high
if (debt_payments $<$ income $\times$ 0.30)	then	debt_low

into an and/or graph.



Transform and/or graph into initial network.

Performance on MONK's problems

	Problem 1	
	Train	Test
Examples Alone	94.35	85.19
CLIPS	100.00	86.34
CLIPS & Examples	91.13	87.04

	Problem 2	
	Train	Test
Examples Alone	95.27	73.61
CLIPS	98.82	68.06
CLIPS & Examples	100.00	81.02

	Problem 3	
	Train	Test
Examples Alone	91.80	75.69
CLIPS	100.00	93.98
CLIPS & Examples	92.62	93.98

Turbine Engine Diagnosis Artificial Neural Network (TEDANN)

Lars J. Kangas[1], Frank L. Greitzer[2], George M. Alexander[2], John A. Sanches[2],
Paul E. Keller[3,4], David D. Turner[4]

[1]Pacific Northwest Laboratory, Applied Physics Center, Computer Science Department
[2]Pacific Northwest Laboratory, Technology Planning and Analysis Center
[3]Pacific Northwest Laboratory, Molecular Science Research Center
[4]Northwest Association of Colleges and Universities in Science (NORCUS)
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-3905 fax: (509) 375-6631 internet: lj_kangas@pnl.gov

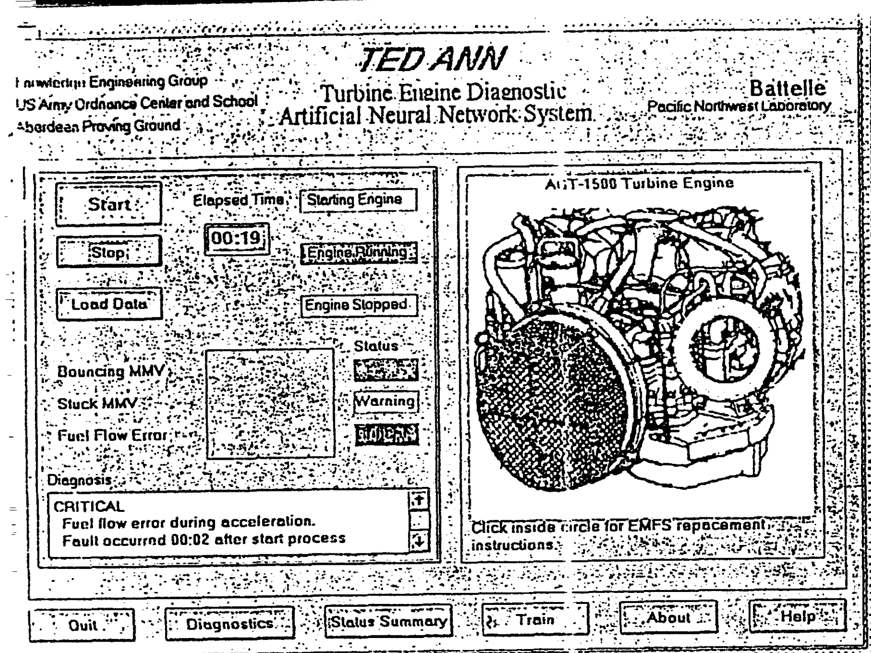
The US Army Ordnance Center and School and Pacific Northwest Laboratory are developing a turbine engine diagnostic system for the M1A1 Abrams tank. This system employs Artificial Neural Network (ANN) technology to perform diagnosis and prognosis of the tank's AGT-1500 gas turbine engine. The initial diagnostic system prototype is referred to as "TEDANN" for Turbine Engine Diagnostic Artificial Neural Networks.

Monitored faults:

1. Bouncing Main Metering Valve
2. Stuck Main Metering Valve
3. Fuel Flow Error (other than type 1 and 2)

Monitored sensor values:

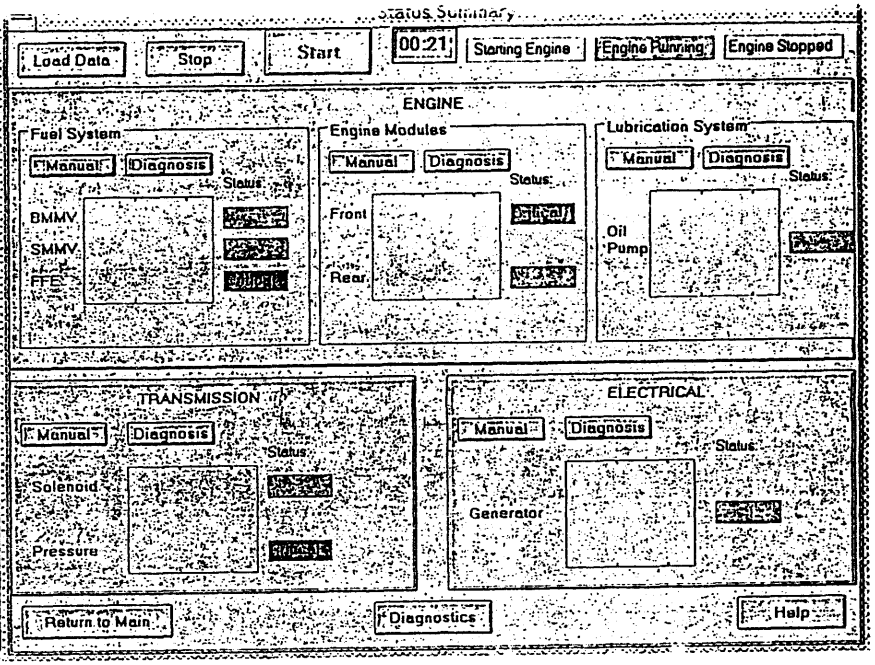
1. Turbine inlet temperature
2. Ambient temperature
3. Compressor Speed
4. Turbine speed
5. Fuel flow requested
6. Actual flow requested
7. Fuel flow solenoid current



Elapsed Time: 00:19

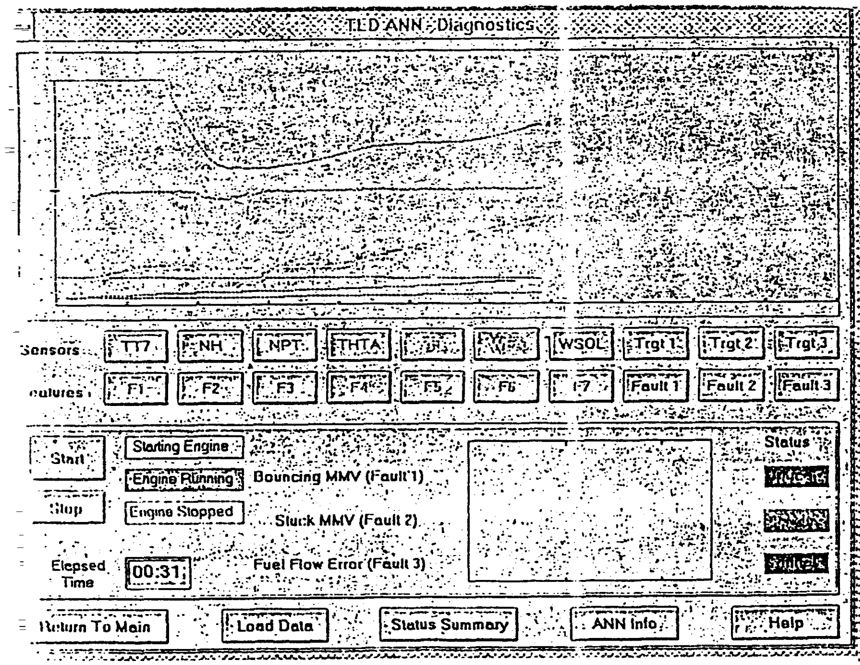
Diagnosis:
CRITICAL
Fuel flow error during acceleration.
Fault occurrnd 00:02 after start process

Figure 1. Display for monitoring and maintaining fuel systems.



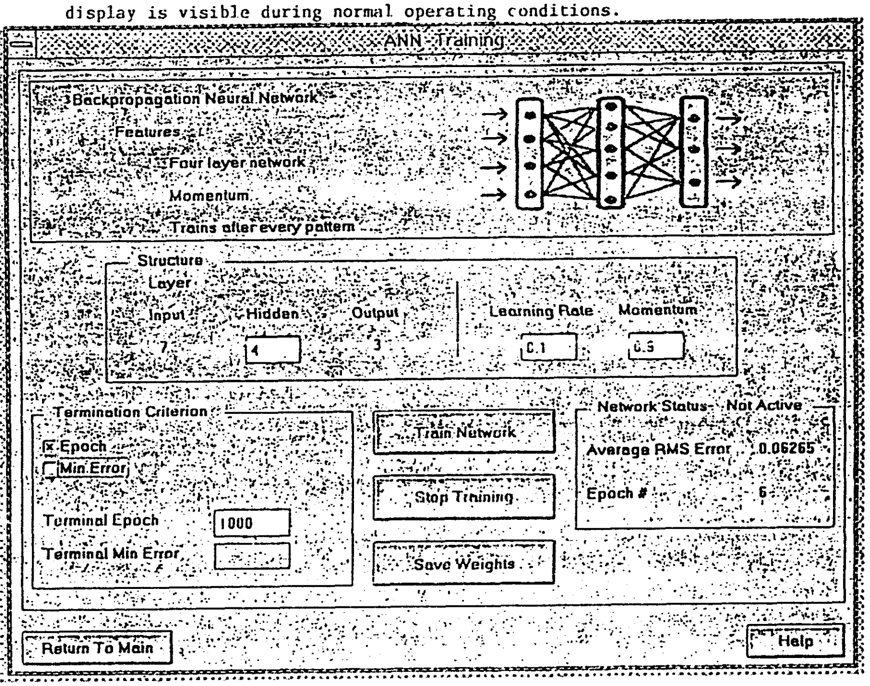
Elapsed time: 00:21

Figure 2. Display for monitoring and maintaining all systems. This display is visible during normal operating conditions.



Elapsed Time: 00:31

Figure 3. Fuel system diagnostic display. The display tracks sensor and neural network information in real-time to aid system development.



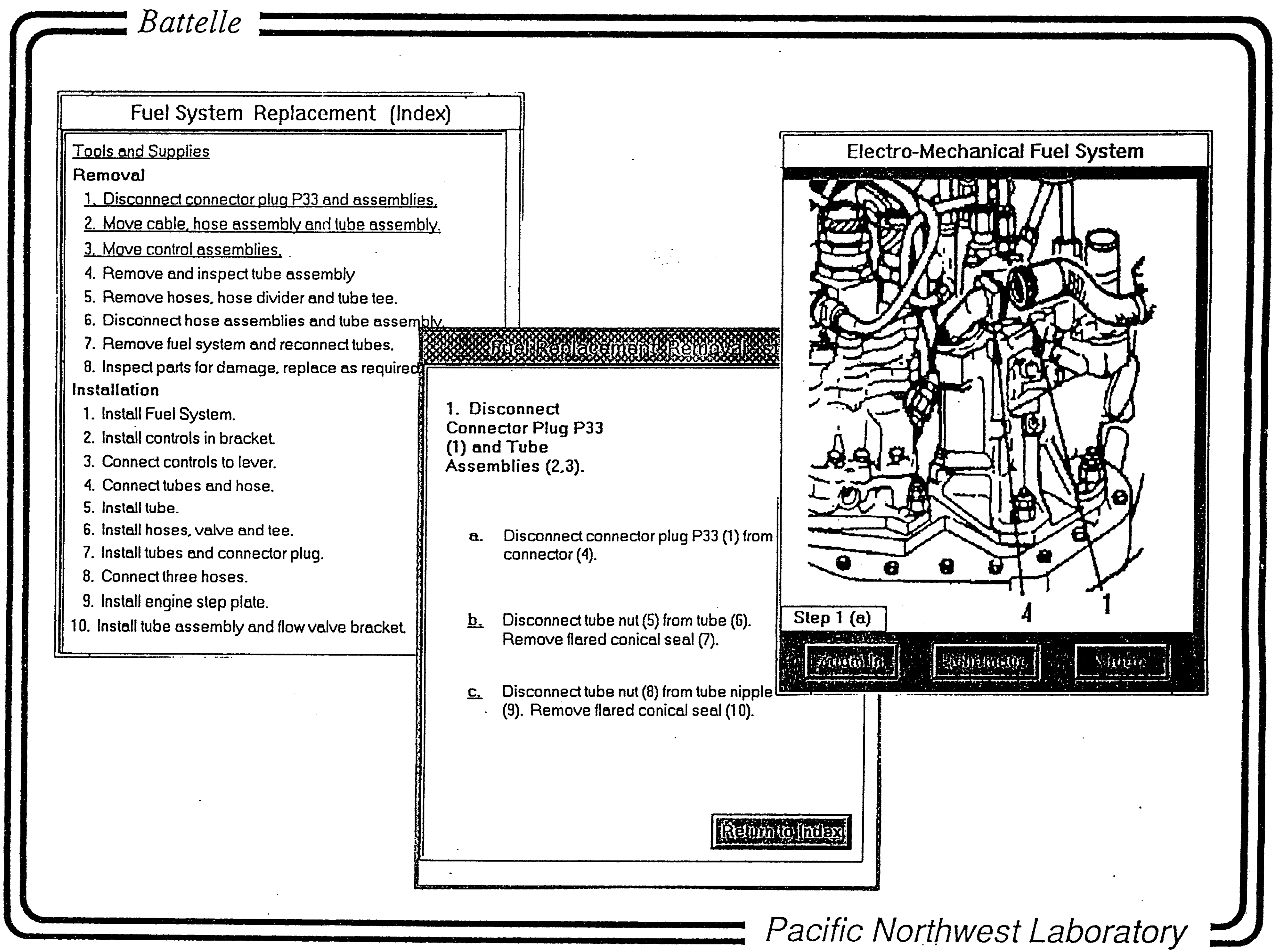
Input	Hidden	Output	Learning Rate	Momentum
7	4	3	0.1	0.5

☒ Epoch
□ Min Error

Terminal Epoch: 1000
Terminal Min Error:

Network Status: Not Active
Average RMS Error: 0.06265
Epoch #: 6

Figure 4. Neural network training display. This display is only used by developers.



Battelle

Fuel System Replacement (Index)

Tools and Supplies

Removal

1. Disconnect connector plug P33 and assemblies.
2. Move cable, hose assembly and tube assembly.
3. Move control assemblies.
4. Remove and inspect tube assembly
5. Remove hoses, hose divider and tube tee.
6. Disconnect hose assemblies and tube assembly.
7. Remove fuel system and reconnect tubes.
8. Inspect parts for damage, replace as required

Installation

1. Install Fuel System.
2. Install controls in bracket.
3. Connect controls to lever.
4. Connect tubes and hose.
5. Install tube.
6. Install hoses, valve and tee.
7. Install tubes and connector plug.
8. Connect three hoses.
9. Install engine step plate.
10. Install tube assembly and flow valve bracket.

Fuel Replacement: Removal

1. Disconnect Connector Plug P33 (1) and Tube Assemblies (2,3).

a. Disconnect connector plug P33 (1) from connector (4).

b. Disconnect tube nut (5) from tube (6). Remove flared conical seal (7).

c. Disconnect tube nut (8) from tube nipple (9). Remove flared conical seal (10).

Return to Index

Electro-Mechanical Fuel System

Step 1 (a)

Pacific Northwest Laboratory

Figure 5. Examples of on-line maintenance pages.

Symbolic Reasoning with Neural Networks

Morgan L. Yim

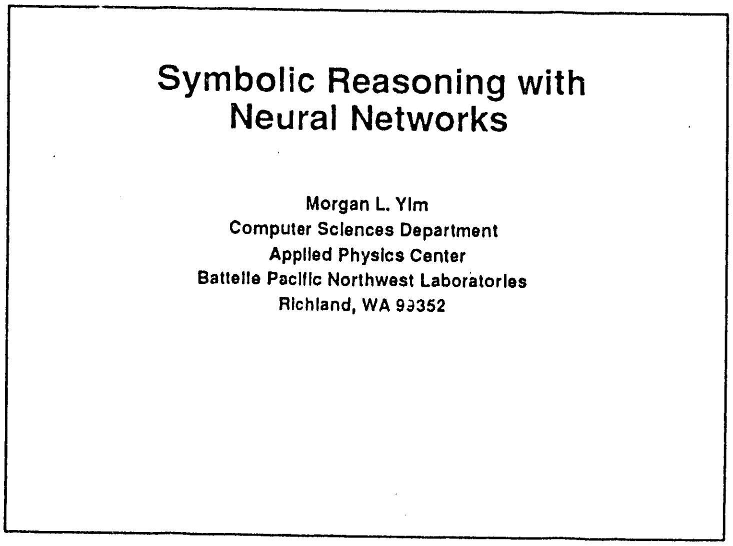
Pacific Northwest Laboratory , Applied Physics Center, Computer Sciences Department
K7-28, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2319 internet: ml_yim@pnl.gov

Rule-based expert systems are symbolic reasoning systems. Many use a form of rules called production rules in an IF-THEN format. These are often based on propositional logic where the conditional parameters take on true or false values. The action clause sets a system state with a true or false value. The limitations of this method can be reduced when confidence factors are utilized within each rule. This addition permits to the rule-based system to reason symbolically about uncertainty, or about incomplete information.

Neural networks can be used in a symbolic fashion which exhibit results similar to expert systems which are constructed by a more complex methodology. Reasoning with incomplete information is automatic with a neural network. Moreover, this approach can be more efficient as compared with a rule-based system. This neural network approach generally evaluated rules in constant time, independent of the number of rules represented. Rule-based production systems generally evaluate rules in linear time as a function of the number of rules in the system.

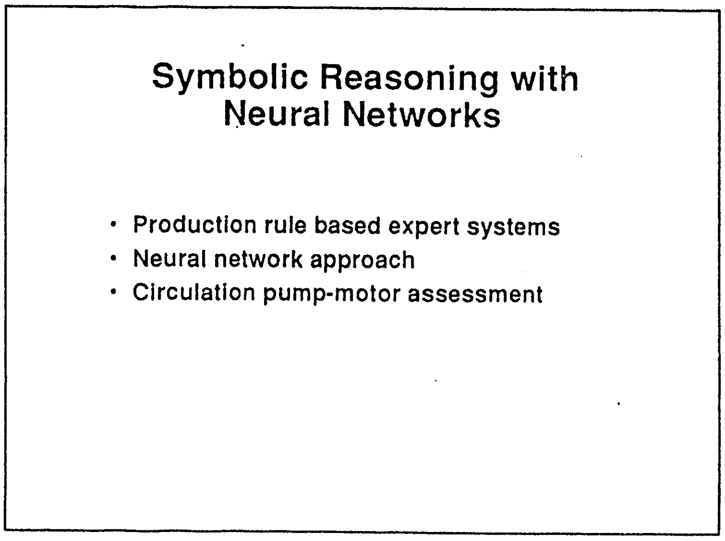
A high-order reasoning system can be constructed using neural networks by the use of an interacting hierarchy of neural clusters. Each neural cluster is capable of solving a portion of a large problem. Each cluster shares its conclusions with other clusters at more abstract levels until a solution is found to the problem.

We have demonstrated the concept of symbolic reasoning with neural networks by using it to identify potential problems with a circulation pump/motor subsystem of a central heating plant. The demonstration of this software will be available in a poster session.



Symbolic Reasoning with
Neural Networks

Morgan L. Yim
Computer Sciences Department
Applied Physics Center
Battelle Pacific Northwest Laboratories
Richland, WA 99352



Symbolic Reasoning with
Neural Networks

- Production rule based expert systems
- Neural network approach
- Circulation pump-motor assessment

I will be talking about the use of neural networks as symbolic reasoners.

I will review how an expert system can be built using a production system.

Then I will show how a neural network can be used to do likewise.

Finally, I will talk about a prototype which uses this method to perform an assessment of circulation pump/motors.

Neural network approach

- **Symbolic representation of domain knowledge**
- **Reasoning objectives are identified**
- **Attributes of objectives are identified**
- **Teach the NN to associate attributes with objectives**
- **One or more objectives can be identified**
- **Precision depends on the availability of attributes**

Neural networks can be used to perform symbolic, rule-style reasoning.

They exhibit results similar to rule-based systems and can be constructed more quickly and simply.

The first step is to identify the reasoning objectives. What do we want the system to conclude?

Next, identify the attributes of the objectives. What does the system look for to support its conclusions?

Then teach the neural network the relationships between the objectives and their attributes.

Reasoning with incomplete information is automatic with a neural network.

With incomplete information, the NN will identify several related objectives.

It will arrive at a single conclusion when sufficient information is present.

Neural network approach

tiger

- has hair
- gives milk
- has pointed teeth
- has claws
- has forward-set eyes
- has black stripes
- has a tawny color

zebra

- has hair
- gives milk
- has hoofs
- has a white color
- has black stripes

Here we want the system to decide between two reasoning objectives, a tiger and a zebra.

Next, we collect the attributes of each animal.

Then we will teach the neural network the relationships between the animals and their corresponding attributes.

While each has attributes which uniquely identifies them, some attributes are shared between them.

Neural network approach

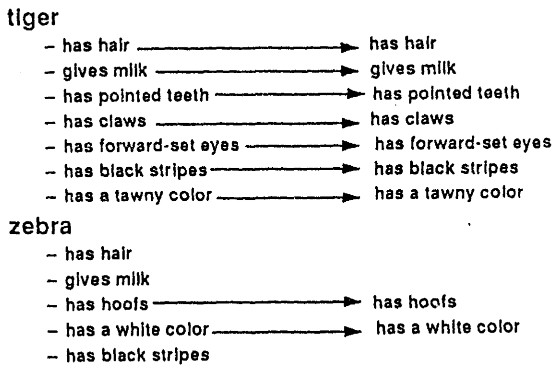
- Simple cross-correlation learning method
- Binary representation of input patterns
- Bipolar representation of learned patterns
- Linear threshold of weighted outputs

Cross-correlation is used to train the neural network.

Bipolar values, 1 and -1, are used to represent the presence or absence of an attribute.

When the neural network is queried, a linear threshold function is used.

Neural network approach



tiger

- has hair → has hair
- gives milk → gives milk
- has pointed teeth → has pointed teeth
- has claws → has claws
- has forward-set eyes → has forward-set eyes
- has black stripes → has black stripes
- has a tawny color → has a tawny color

zebra

- has hair
- gives milk
- has hoofs → has hoofs
- has a white color → has a white color
- has black stripes

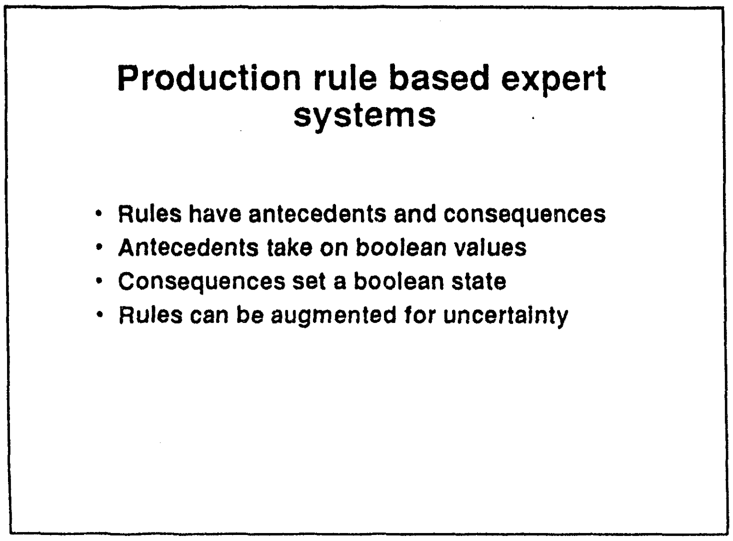
In preparation to train the neural network, all the attributes are combined into a single list.

All duplicates are removed, so there is a single occurrance for any attribute.

This defines the input nodes of the neural network.

The number of objectives, the animals, defines the output nodes.

In this case, we have an neural network having 9 input nodes and 2 output nodes.



Production rule based expert systems

- Rules have antecedents and consequences
- Antecedents take on boolean values
- Consequences set a boolean state
- Rules can be augmented for uncertainty

Rule-based expert systems are symbolic reasoning systems.

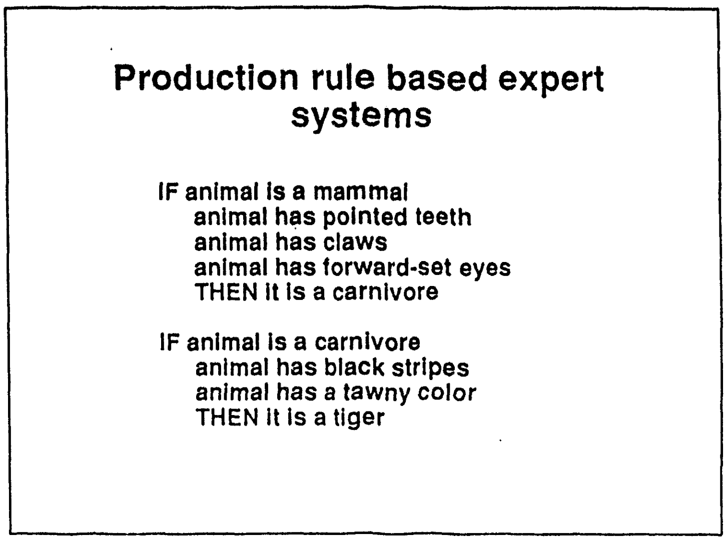
Many use a form of rules called production rules in an IF-THEN format.

These are often based on propositional logic where the conditional parameters take on true or false values.

The action clause likewise, set a system state with a true or false value.

The limitations of this method can be reduced when confidence factors are utilized within each rule.

This addition permits to the rule-based system to reason symbolically about uncertainty, or about incomplete information.



Production rule based expert systems

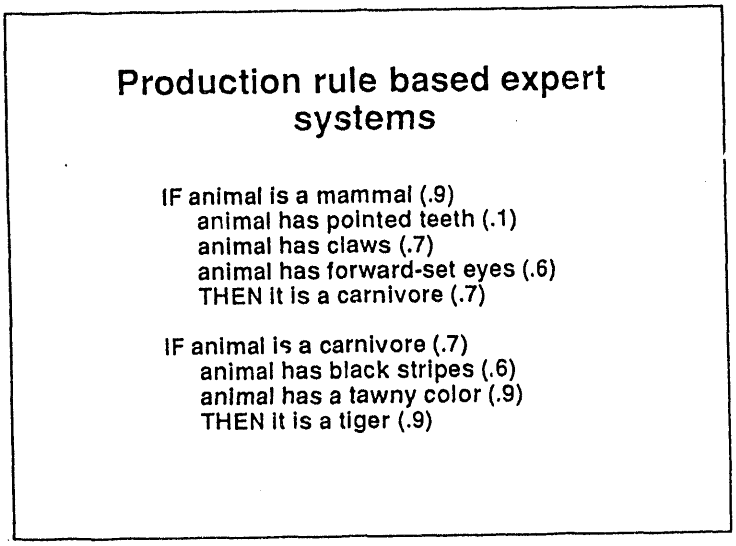
IF animal is a mammal
animal has pointed teeth
animal has claws
animal has forward-set eyes
THEN it is a carnivore

IF animal is a carnivore
animal has black stripes
animal has a tawny color
THEN it is a tiger

This is an example what one might see in a expert system.

The conclusion part of the rule is true when all the conditions are true.

The conclusion is then used in another rule as a precondition for considering additional information.



Production rule based expert systems

IF animal is a mammal (.9)
animal has pointed teeth (.1)
animal has claws (.7)
animal has forward-set eyes (.6)
THEN it is a carnivore (.7)

IF animal is a carnivore (.7)
animal has black stripes (.6)
animal has a tawny color (.9)
THEN it is a tiger (.9)

When only incomplete information is available, the rules of an expert system can be augmented with numeric values.

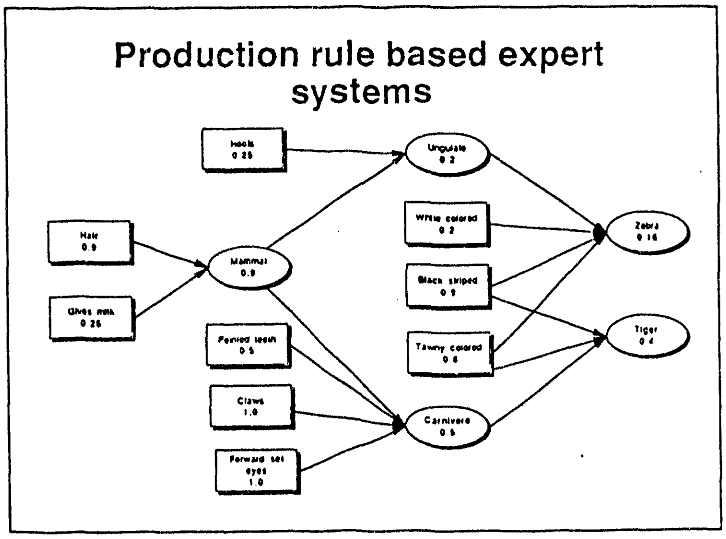
Sometimes statistical techniques are used, like Baysian or Dempster-Shaeffer.

Sometimes a confidence factor is used.

In any case, some method for combining the numeric value is used to determine a total value for the rule.

Then a comparison is made to determine whether the given rule can fire.

When fired, the rule not only sets a new system state but also propagate the conclusion value to subsequent rules.



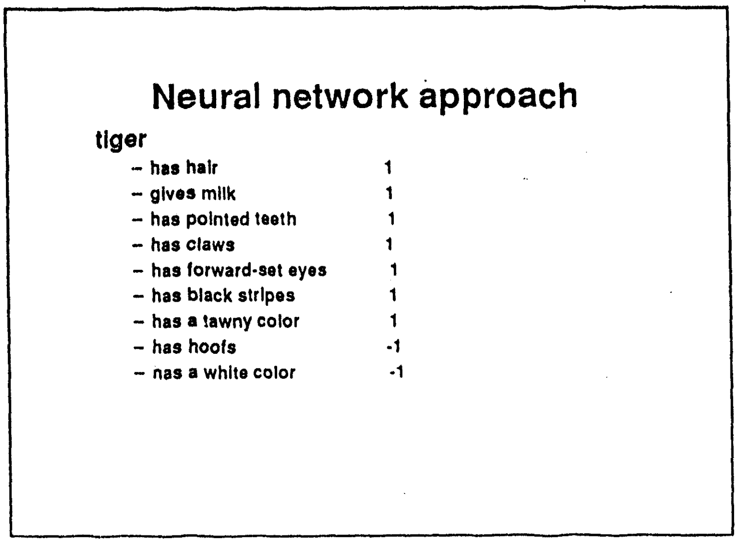
Production rule based expert systems

This is an illustration of a rules network.

Here we can see graphically how confidence factors propagate from rule to rule to arrive at a conclusion.

The rectangles indicate information coming from outside the system.

The ovals indicate derived information from rule firings.



Neural network approach

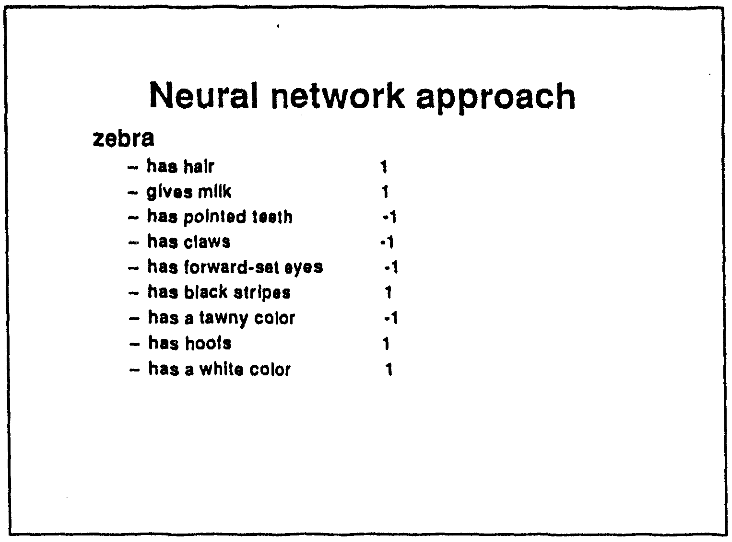
tiger

– has hair	1
– gives milk	1
– has pointed teeth	1
– has claws	1
– has forward-set eyes	1
– has black stripes	1
– has a tawny color	1
– has hoofs	-1
– nas a white color	-1

This is the training pattern for recognizing a tiger.

Given complete set of attributes, the 1s indicate the presence of tiger attributes.

The -1s indicate the attributes which do not apply to the tiger.



Neural network approach

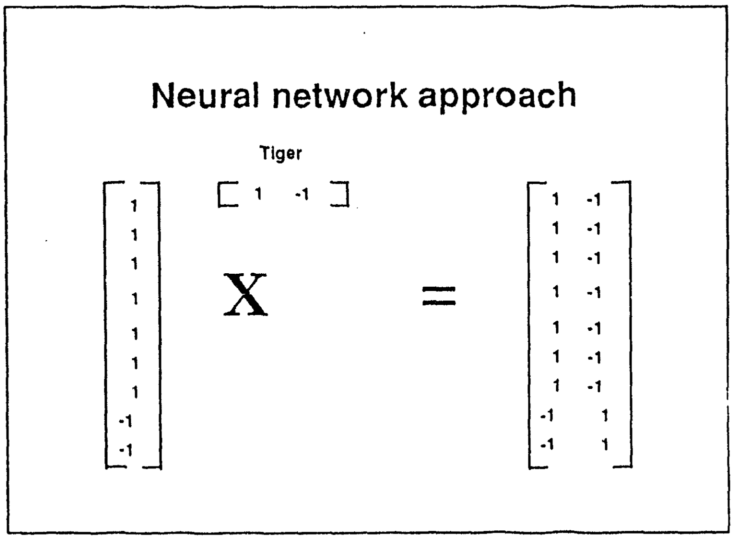
zebra

– has hair	1
– gives milk	1
– has pointed teeth	-1
– has claws	-1
– has forward-set eyes	-1
– has black stripes	1
– has a tawny color	-1
– has hoofs	1
– has a white color	1

This is the training pattern for recognizing a zebra.

Here we notice that the zebra shares some attributes with the tiger.

Other tiger attributes clearly do not apply to the zebra.



Neural network approach

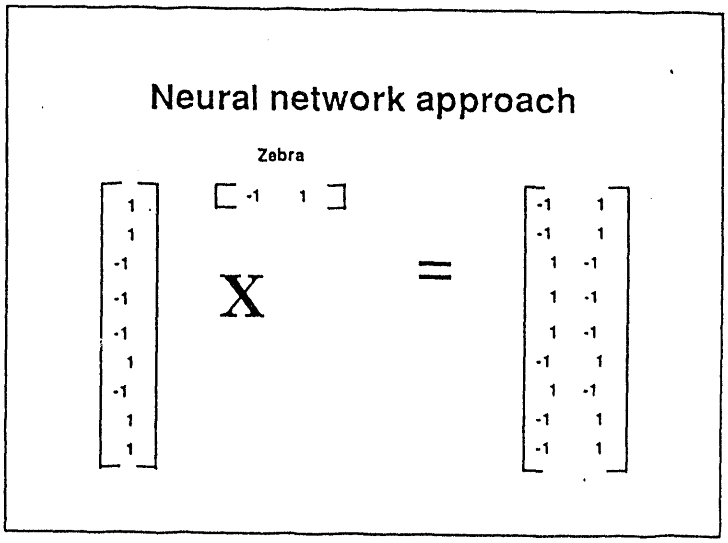
Tiger

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 \end{bmatrix} \quad \mathrm{X} \quad = \quad \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

A cross correlation is made with the tiger training vector and a vector representing the two animals.

The tiger position is set to 1 while the zebra is set to -1.

The result is a matrix which represents the correlation of the tiger's attributes with the tiger.



Neural network approach

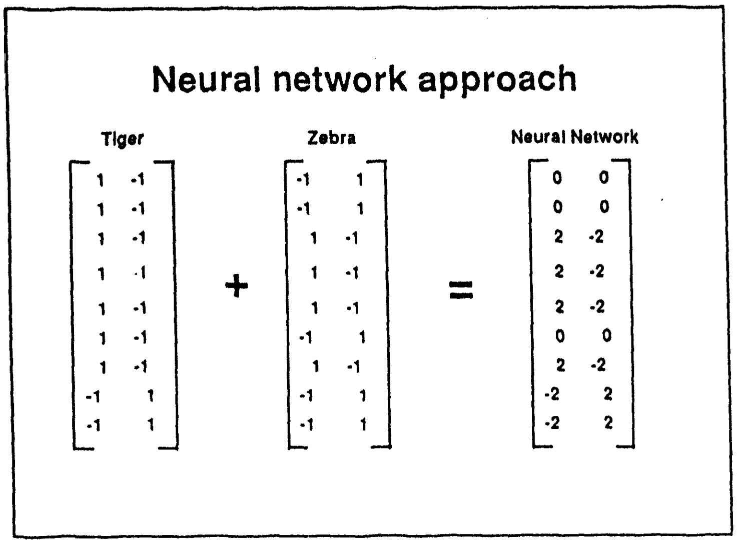
Zebra

$$\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \end{bmatrix} \quad \mathrm{X} \quad = \quad \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Similarly, the pattern for the zebra is cross correlated.

This time, the tiger's position is set to -1 and the zebra's to 1.

The result is a matrix which represents the correlation of the zebra's attributes with the zebra.

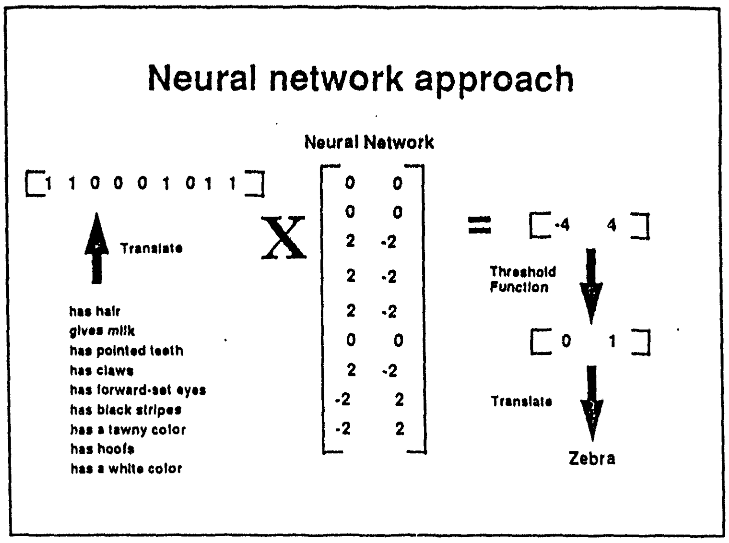


Neural network approach

$$\begin{array}{ccccc} \text{Tiger} & & \text{Zebra} & & \text{Neural Network} \\ \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} & + & \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} & = & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & -2 \\ 2 & -2 \\ 2 & -2 \\ 0 & 0 \\ 2 & -2 \\ -2 & 2 \\ -2 & 2 \end{bmatrix} \end{array}$$

The two resulting matrices are added together to form the neural network.

The NN is now ready for use.



Neural network approach

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \mathbf{X} \overset{\text{Neural Network}}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & -2 \\ 2 & -2 \\ 2 & -2 \\ 0 & 0 \\ 2 & -2 \\ -2 & 2 \\ -2 & 2 \end{bmatrix}} = \begin{bmatrix} -4 & 4 \end{bmatrix} \xrightarrow{\text{Threshold Function}} \begin{bmatrix} 0 & 1 \end{bmatrix} \xrightarrow{\text{Translate}} \text{Zebra}$$

Translate

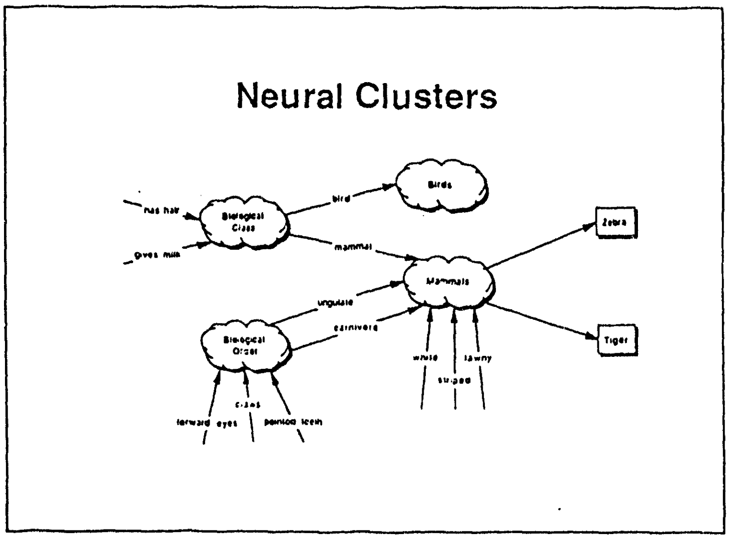
has hair
gives milk
has pointed teeth
has claws
has forward-set eyes
has black stripes
has a tawny color
has hoofs
has a white color

Binary values, 0 and 1 are used to represent attribute observations.

These observations are represented in a vector and correlated with the NN.

The result from the NN is a vector representing the strength of correlation between the observed attributes and the learned attributes.

By picking the largest value with a threshold function, the NN can indicate its best conclusion given the provided information.



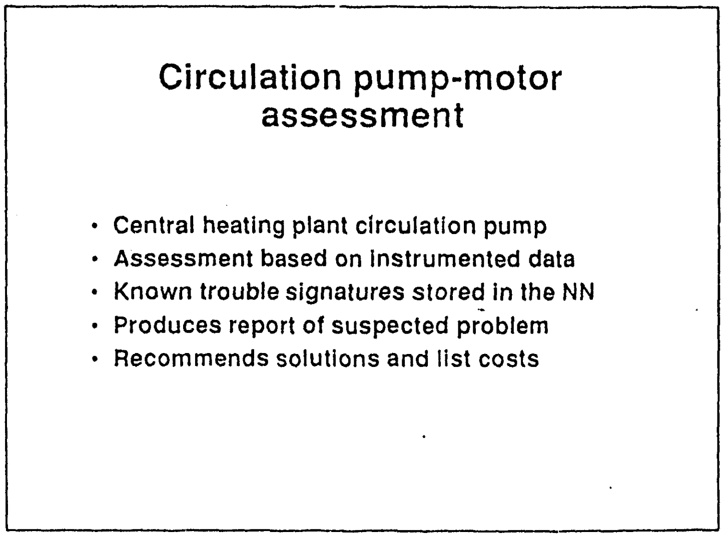
Neural Clusters

The neural network just described could be thought of as a neural cluster.

A cluster is a neural network capable of solving a subproblem.

A cluster can propagate its conclusions to other clusters to contribute to the total solution.

A neural lattice is a collection of neural clusters arranged in a cooperative fashion to subdivide and to solve a large problem.



Circulation pump-motor assessment

- Central heating plant circulation pump
- Assessment based on instrumented data
- Known trouble signatures stored in the NN
- Produces report of suspected problem
- Recommends solutions and list costs

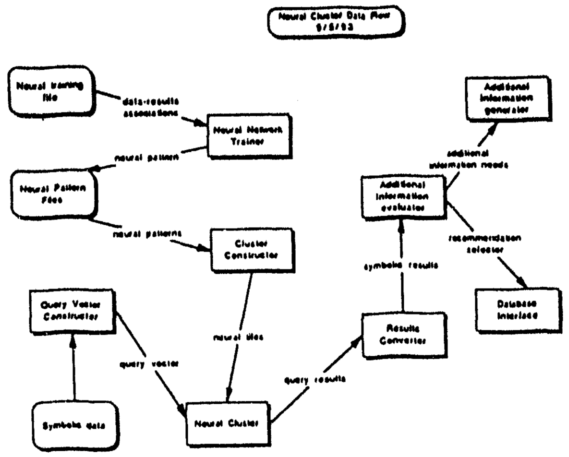
One such large problem is the diagnosis of a circulation pump/ motor subsystem of a central heating plant.

This subsystem is instrumented in such a way that data for essential attributes are collected.

A neural network is used to monitor for known trouble signatures.

When trouble is detected, a report is generated about the suspected problems.

The solutions and cost for each are listed for decision-making.



Circulation pump-motor assessment

- **Detects problem mechanisms in pumps**
- **Recognition of problem signatures**
- **Symbolic representation of signatures**
- **Numeric data pre-processed into symbols**
- **Prototype demonstration available**

By the use of neural network as a symbolic reasoner, we were able to detect problem mechanisms in circulation pump/motors.

This was accomplished through the recognition of problem signatures by the neural network.

The signatures were represented symbolically for input into the neural network.

The signatures relate to the numeric values taken from measuring instruments attached to the pump/motor.

Finally, we have available the prototype software for the pump/motor application.

Please contact me for a demonstration.

Verification and Validation of Neural Networks

James S. Dukelow, Jr.

Battelle Pacific Northwest Laboratories, Engineering Technology Center
K8-37, P.O. Box 999, Richland, WA 99352
phone: (509) 372-4072 internet: js_dukelow@pnl.gov

This talk considers the intersection between the software engineering process and the design, training, and testing of artificial neural networks. This common ground is of interest because neural networks offer a number of capabilities of interest for nuclear industry applications, including safety-related applications, but the process of developing and testing a neural network does not appear to fit well into the software engineering process as implemented in the existing body of software standards.

An artificial neural network consists of a collection of generally identical processing elements, the artificial neurons, each with input and output lines. For purposes of specificity, we can consider the artificial neurons to be arranged in a layered architecture, with an input layer receiving information from the outside world, multiplexing it, and passing it on to a "hidden" processing layer, which itself passes processed data on to an output layer for further processing and output to the outside world. The layers are generally richly connected (i.e., each neuron in a given layer is connected to each neuron in the adjacent layers) and the connecting lines have associated connection "weights", which can be modified during the training or "learning" process. Under appropriate assumptions, a neural network can be trained to learn the relationship between a set of input/output pairs (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), called the "training set", where each x_i is an element of k-dimensional Euclidean space, R^k, and each y_i is an element of R^m. For instance, we could have each $y_i = f(x_i)$ for some function f mapping R^k into R^m. Detailed, mathematically-oriented descriptions of various neural network architectures, "learning rules", and applications can be found in Ref. 1.

Neural networks have been developed and used successfully for applications in pattern recognition, optimization, approximation, adaptive filtering, data fusion, machine vision, voice recognition, and process control. Recent years have shown increasing interest in the nuclear industry in potential applications of neural networks to industry needs. An indication of this interest can be found in the proceedings of the 1992 Workshop on Neural Network Computing for the Electric Power Industry, cosponsored by the Electric Power Research Institute and the International Neural Network Society (Ref. 2). Precisely because of verification and validation concerns in the regulated environment of the nuclear industry, all of the fielded applications of neural networks to electric utility industry problems are limited to fossil facilities or to the transmission and distribution network.

For several reasons, verification and validation of neural networks are seen as possible impediments to nuclear industry applications, particularly safety-related applications. For many existing applications, the neural network development has had a distinctly empirical or experimental flavor; considerable judgment may be required in the choices of the neural network architecture, the "training data set", the "training test data set", the "validation data set", and the stopping point for the training effort. The neural network development process is significantly different from the software development process for "normal" procedural software, and is not well-supported by existing software consensus standards. It is normal for many fielded neural net applications to give the wrong answer some fraction of the time, a situation violating the spirit, if not the letter of existing standards. This presentation will attempt to expand on the thoughtful, qualitative discussion of some of the issues of verification and validation of neural networks found in Ref. 3.

The behavior of a neural network architecture is generally determined by the transfer functions of its neurons (i.e., the mapping of their input into their output), by the learning rule, and by the architecture, itself (i.e., by the way in which the neurons are interconnected). Numerous recent results in the literature have been able to describe this behavior as a mathematical theorem on the existence and/or construction of an approximation to a member of some appropriately defined function space, that is, as a result in approximation or optimization theory. This presentation will discuss some of these results, including the Kolmogorov Mapping Theorem and various Universal Approximation Theorems (Refs. 1 and 4).

The bottom line question: Are adequate V&V methods available to justify use of neural networks in safety-critical applications? It is the author's conviction that a combination of the methods discussed in the Peterson report (Ref. 3) with the recently-developing theoretical basis for characterizing neural network behavior form a potential framework for rigorous justification of safety-related applications of neural networks. In addition, it will be important that the body of consensus standards not rule out neural networks by virtue of inflexible application of rules more appropriate to the environment of procedural software in which the standards were developed.

References

1. Hecht-Nielsen, Robert, Neurocomputing, Addison-Wesley, Reading, Massachusetts, 1990.

2. Sobajic, Dejan J. (Ed.), 1993, Neural Network Computing for the Electric Power Industry: Proceeding of the 1992 INNS Summer Workshop, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

3. Peterson, Gerald, 1992, "A Framework for Neural Network Evaluation", McDonnel Douglas Technical Report MDRL TN-92-02, McDonnel Douglas Research Laboratories, St. Louis, Missouri.

4. Ito, Yoshifusa, 1992, "Approximation of Continuous Functions on R^d by Linear Combinations of Shifted Rotations of a Sigmoid Function With and Without Scaling", in Neural Networks, Vol. 5, No. 1, pp. 105-115, Pergamon Press, New York.

VERIFICATION & VALIDATION OF NEURAL NETWORKS

J. S. Dukelow, Jr.

Presented at the

Neural Network Workshop for the Hanford Community

January 26, 1994

- Context for this talk
- Usual Software Quality Assurance Requirements
- Usual Neural Network V & V
- Peterson's Approach
- Stacked Generalizers
- Probabilistic Proofs
- Mathematical Properties
- Conclusions

CONTEXT FOR THIS TALK

- IEEE 1012 Working Group
- Absolute SQA Requirements
- SQA for "Non-Testable" Software
- Application to V & V of Neural Networks

USUAL SOFTWARE QA REQUIREMENTS

- Software Development Process Control
- Testability of all Software Requirements
- Testing exercises all of Input Space
- Testing exercises all Control Paths

Interference and Sequential Training of Connectionist Networks: What can we do?

A. Lynn Franklin

University of Washington, Department of Psychology
Seattle, WA
phone: (206) 543-6695 internet: alfrank@u.washington.edu
and
Pacific Northwest Laboratory , Technology, Planning, and Analysis Center
K8-17, P.O. Box 999, Richland, WA 99352

Connectionist networks have been successfully used to perform a wide variety of information processing tasks. While these accomplishments may be impressive, each has been required to submit to two fundamental constraints. First, the configurations chosen to implement each network are generally selected through a less than sophisticated trial and error process. A second, and more important, issue is that each of these networks must be trained in a repetitive sweep fashion with all known examples being presented. The introduction of new examples generally requires retraining on all previously learned examples, along with the new example. Without retraining, connectionist networks are susceptible to interference (degradation of performance) on previously learned examples. This dependence on sweep learning, which is not typical for humans, indicates existing learning algorithms have yet to capture the learning mechanisms of biological neural networks.

This paper discusses one approach for training connectionist networks, suppressive specialization, that addresses both of these issues. The basic concept of suppressive specialization will be presented along with several case studies demonstrating the impact of this approach on interference. In addition, several issues that result from the ability to use sequential training will be introduced.

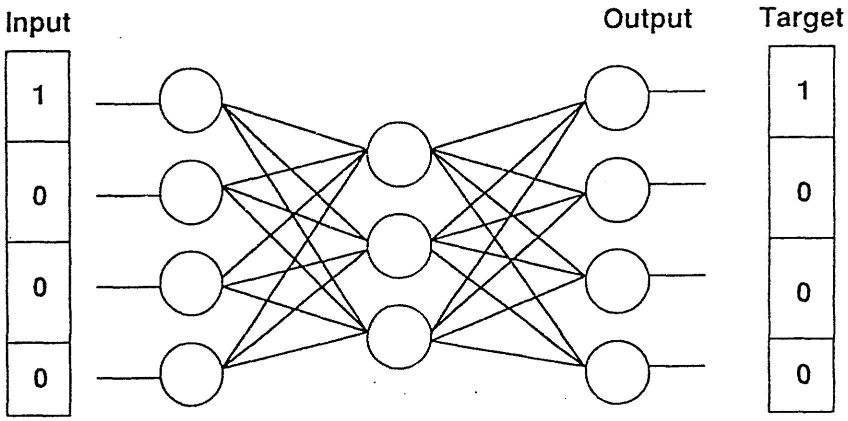
Agenda

- objectives, motives, and definitions
- examples of interference
- suppressive specialization
- demo

Objectives

- Interference/Sequential Training
- Selection of Hidden Layer Units

Four Bit Encoder



Initial Training

Training Set	Evaluate	Output After Training			
1 0 0 0	1 0 0 0	0.97	0.04	0.04	0.04
0 1 0 0	0 1 0 0	0.04	0.97	0.04	0.04
0 0 1 0	0 0 1 0	0.04	0.04	0.97	0.04

Sequential Training

Training Set	Evaluate	Output After Training			
1 0 0 0	1 0 0 0	0.97	0.04	0.04	0.04
0 1 0 0	0 1 0 0	0.04	0.97	0.04	0.04
0 0 1 0	0 0 1 0	0.04	0.04	0.97	0.04
0 0 0 1					

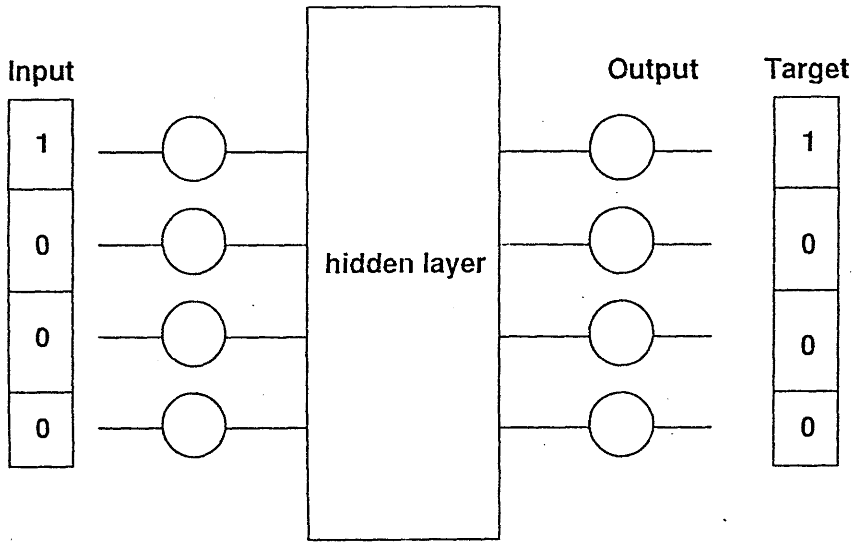
Following Sequential Training

Training Set	Evaluate	Output After Training			
1 0 0 0	1 0 0 0	0.97	0.04	0.04	0.04
0 1 0 0	0 1 0 0	0.04	0.97	0.04	0.04
0 0 1 0	0 0 1 0	0.04	0.04	0.97	0.04
0 0 0 1	1 0 0 0	0.97	0.04	0.04	0.04
	0 1 0 0	0.04	0.97	0.04	0.04
	0 0 1 0	0.04	0.04	0.97	0.04
	0 0 0 1	0.04	0.04	0.04	0.97

Ratcliff Performance

Training Set	Evaluate	Output After Training			
1 0 0 0	1 0 0 0	0.94	0.04	0.05	0.02
0 1 0 0	0 1 0 0	0.05	0.94	0.04	0.02
0 0 1 0	0 0 1 0	0.04	0.05	0.94	0.02
0 0 0 1	1 0 0 0	0.77	0.01	0.01	0.90
	0 1 0 0	0.01	0.75	0.01	0.92
	0 0 1 0	0.01	0.01	C.73	0.92
	0 0 0 1	0.03	0.04	0.04	0.95
	1 1 1 1	0.02	0.03	0.04	0.95
	0 1 1 0	0.00	0.36	0.33	0.91

Four Bit Encoder



Suppressive Specialization

- Hidden Layer Pool
- Lateral Inhibition
- Plasticity Adjustment
- Cooperation Function

Suppressive Specialization

Input	Evaluate	Output After Training			
1 0 0 0	1 0 0 0	0.97	0.01	0.02	0.02
0 1 0 0	0 1 0 0	0.01	0.95	0.01	0.01
0 0 1 0	0 0 1 0	0.01	0.01	0.97	0.02
0 0 0 1	1 0 0 0	0.74	0.01	0.01	0.01
	0 1 0 0	0.01	0.58	0.01	0.01
	0 0 1 0	0.02	0.01	0.81	0.02
	0 0 0 1	0.02	0.01	0.01	0.95
	1 1 1 1	0.70	0.57	0.94	0.84
	0 1 1 0	0.01	0.74	0.91	0.02

Ground Penetrating Radar Target Recognition

Lars J. Kangas, Gerald A. Sandness, Shawn J. Bohn

Pacific Northwest Laboratory, Applied Physics Center, Computer Science Department
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-3905 fax: (509) 375-6631 internet: lj_kangas@pnl.gov

A feasibility study was performed for Osaka Gas Company, Japan using Artificial Neural Networks (ANNs) to recognize underground utility pipes from images generated by ground penetrating radar.

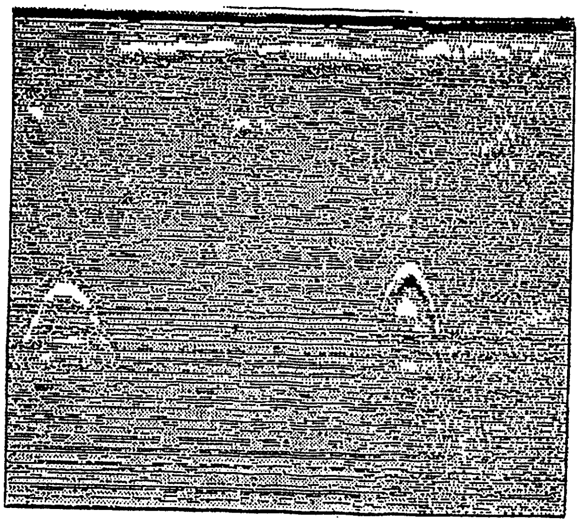


Figure 1. Ground penetrating radar image. An image is achieved by viewing a sequence of individual radar ping returns side-by-side. The vertical dimension of the image is depth in the ground (approximately 8 ft), with the surface on the top. The horizontal dimension represents the run along a line on the surface (approximately 30 ft).

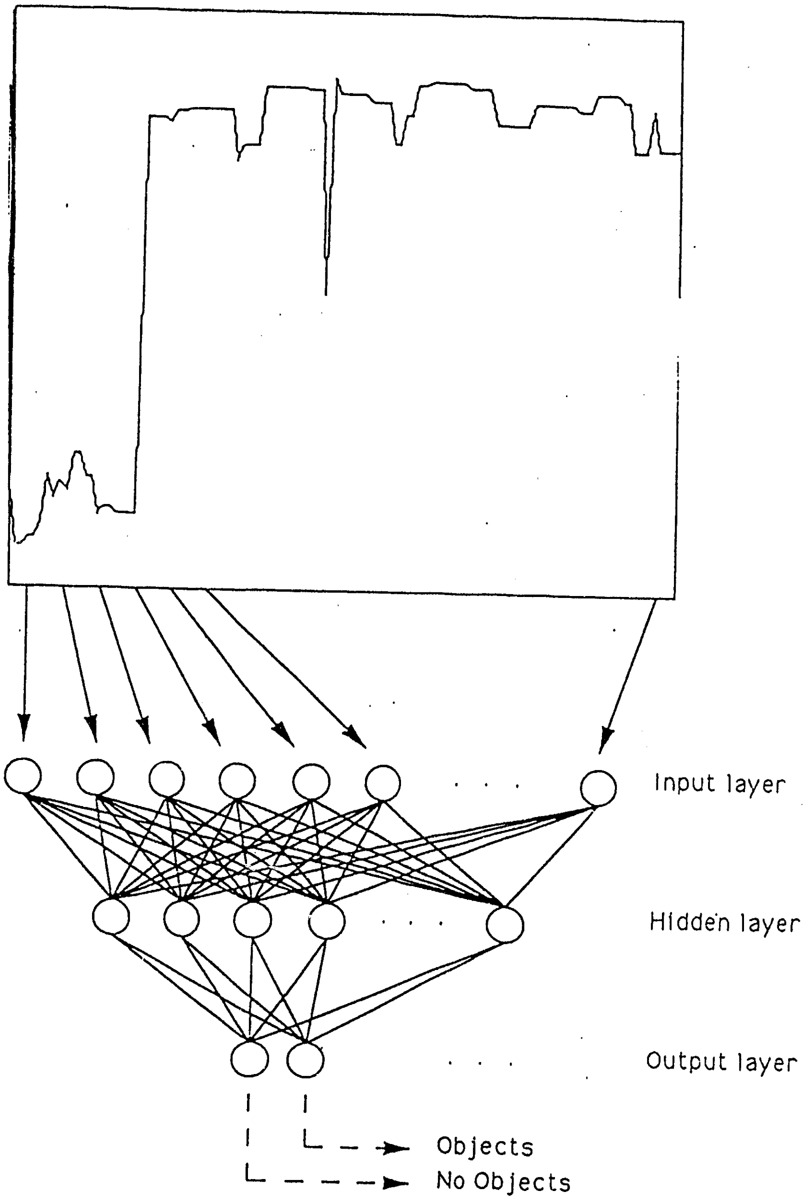


Figure 2. Each individual radar ping is analyzed by an ANN for pipes. The input to the ANN is the peak-to-peak values in the phase component of a Fourier transformed radar ping. The ANN output determines whether there is a pipe in the ping.

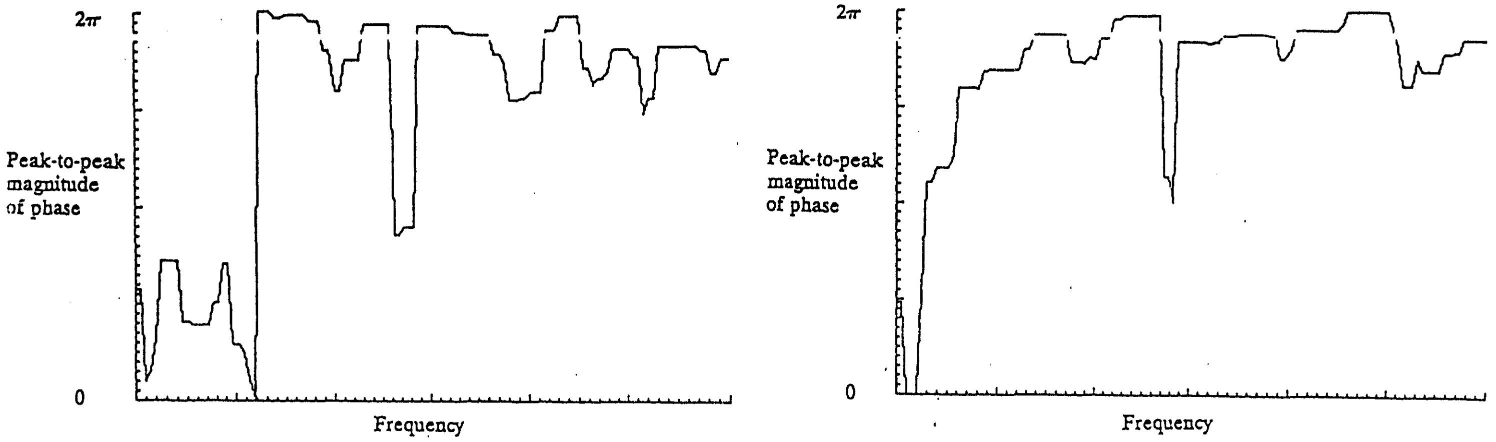


Figure 3. Peak-to-peak magnitudes in the phase components of Fourier transformed radar pings. The left and right graphs are pings without and with pipes, respectively. The ANN uses the differences in these graphs to determine whether there is a pipe in the ping.

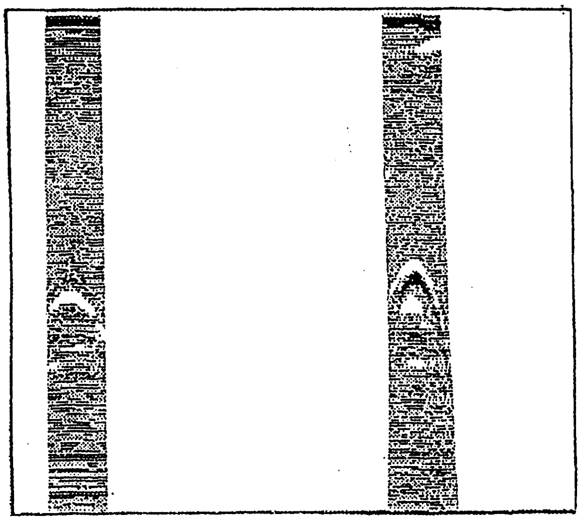


Figure 4. Radar image showing the pings which were determined to have signatures of pipes in the ANN in Figure 2. Only these pings will be further processed for pipe recognition.

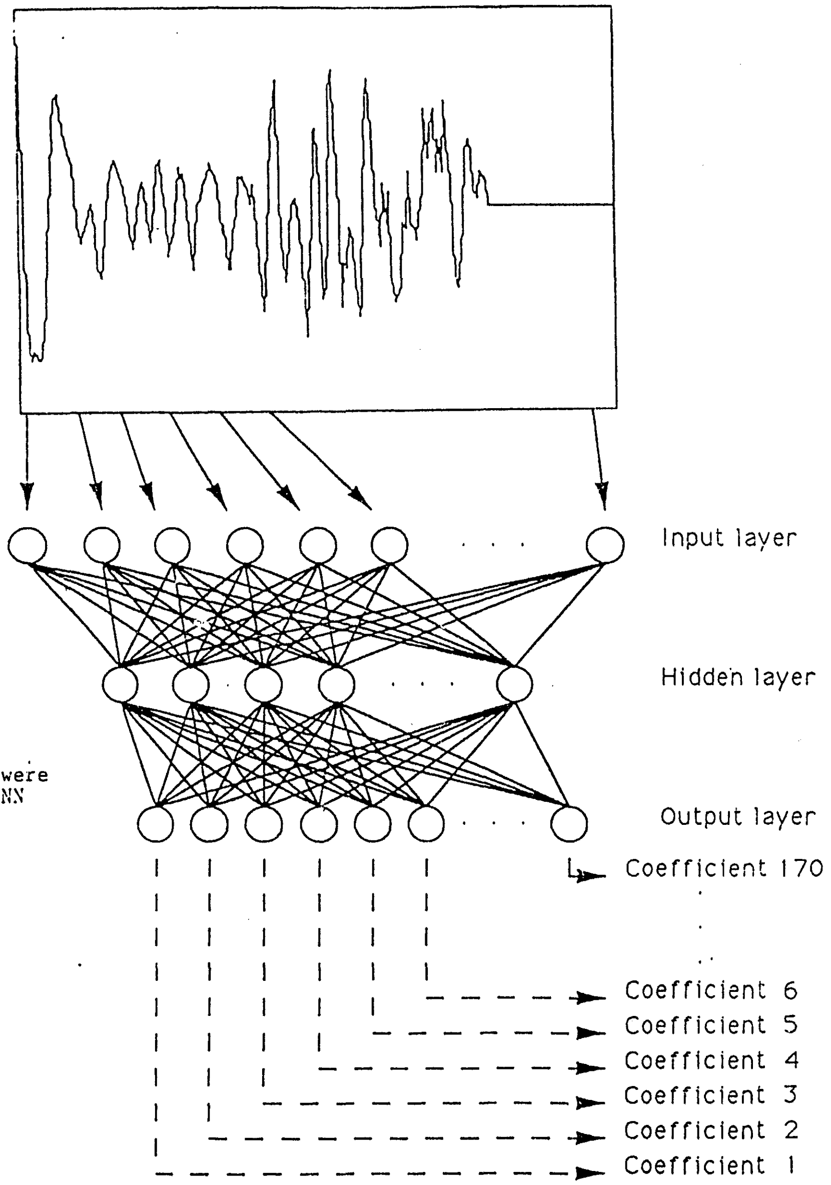


Figure 5. A radar ping is analyzed in an ANN to determine at which depth in the ping there is a signature from a pipe. The ping signal is sampled into 170 coefficients. The ANN output is the recognition of a pipe signature at every coefficient. This scheme allows multiple pipes to be recognized in a single ping.

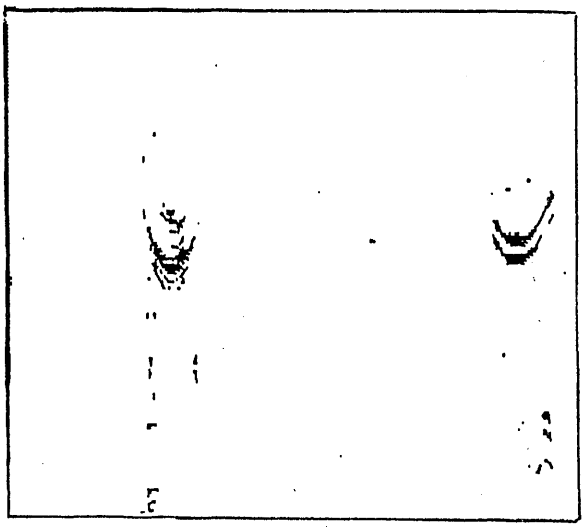


Figure 6. Image reproduced from the output of the ANN in Figure 5. The dark colors represent recognition of pipe signatures.

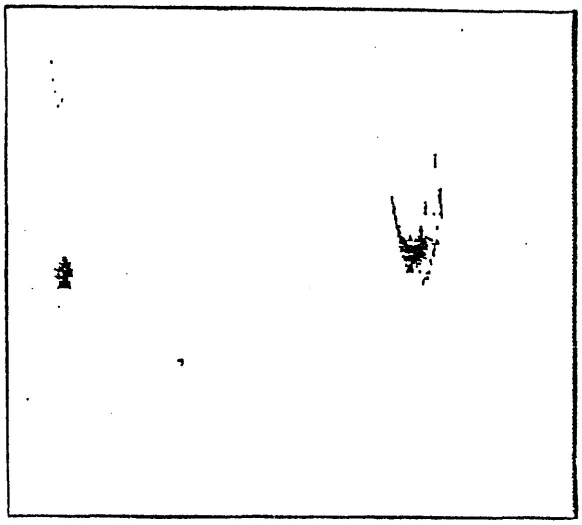


Figure 7. The image in Figure 6 after synthetic aperture processing.

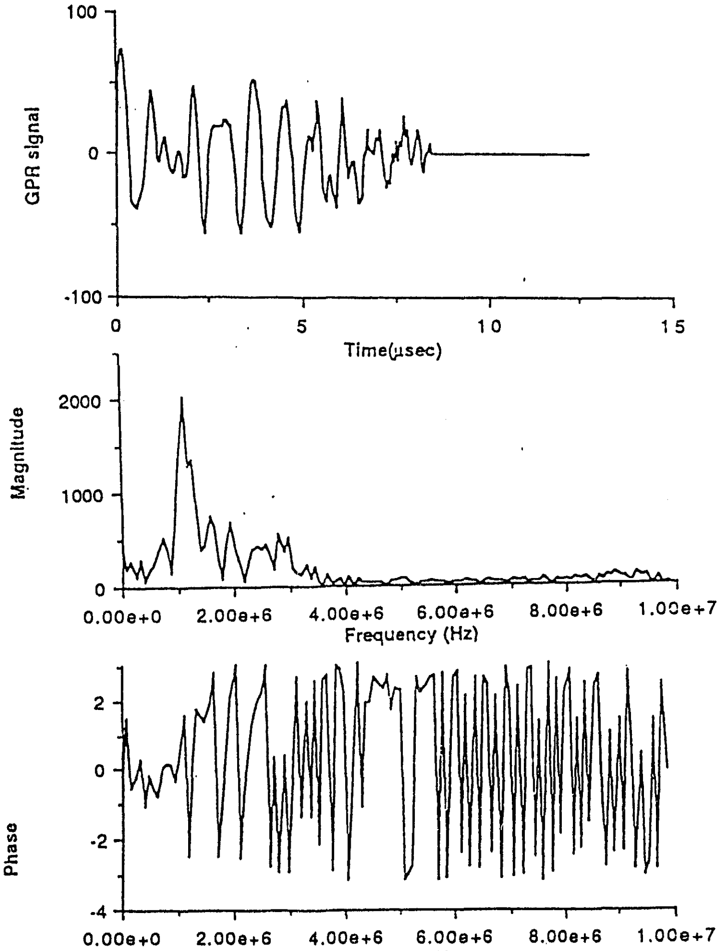


Figure 8. The three graphs show the unprocessed radar ping, the magnitude and phase components of the Fourier transformed radar ping.

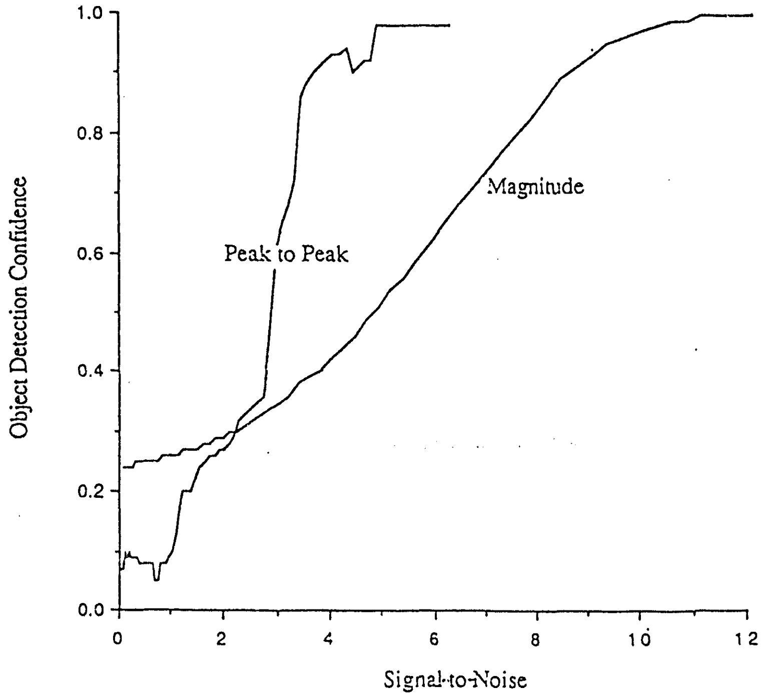


Figure 9. The graph shows how confident the ANN system is in recognizing a pipe at different signal-to-noise ratios. The graph shows results from using both the peak-to-peak magnitudes of the phase and the magnitude components of Fourier tranformed radar pings.

Neural Network Applications in the Environmental and Molecular Sciences Laboratory (EMSL)

Paul E. Keller, Richard T. Kouzes, Lars J. Kangas

Pacific Northwest Laboratory, Molecular Science Research Center,
Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov

The construction of the Environmental and Molecular Sciences Laboratory (EMSL) at the Pacific Northwest Laboratory is about to begin. This facility will assist in the overall environmental restoration and waste management mission at the Hanford Site by providing basic and applied research support. This poster identifies several applications in the Environmental and Molecular Sciences Laboratory where neural network solutions can potentially be beneficial. These applications include real-time sensor data acquisition and analysis, spectral analysis, process control, theoretical modeling, and data compression.

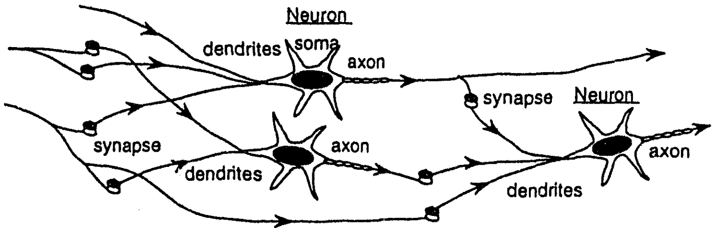
Mission of the EMSL

The Environmental and Molecular Sciences Laboratory (EMSL) will be an essential part of our ability to provide the necessary technology solutions to achieve environmental restoration objectives, including

- developing new technologies to solve environmental restoration and waste operations problems
- developing technology to improve environmental restoration and waste management effectiveness, efficiency, and safety by establishing the scientific basis for contaminant transport in groundwater, surface water, soil, and atmospheric systems
- enhancing educational programs and initiatives by providing the necessary capabilities to support cooperative educational and training programs with academic institutions and DOE national laboratories
- encouraging collaboration and technology transfer among Federal agencies, State and local governments, industry, academia, and the international community by operating as a national DOE user facility open to all scientists.

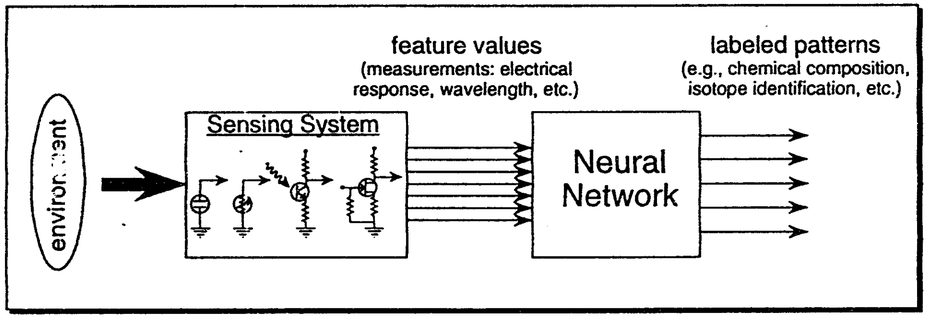
Neural Network Fundamentals

Simple Model of a Neural Network

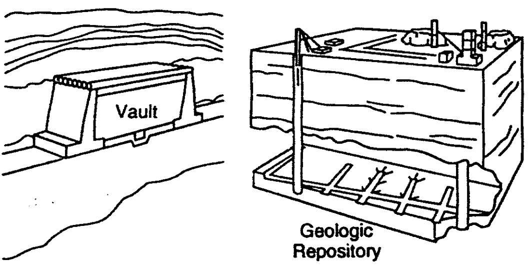


neurons ⇔ nonlinear processing elements
synapses ⇔ weighted interconnections between neurons
dendrites/axons ⇔ communication channels
neural network ⇔ system of highly interconnected, nonlinear processors working in unison

Sensor Data Acquistion and Analysis


feature values
(measurements: electrical response, wavelength, etc.)
labeled patterns
(e.g., chemical composition, isotope identification, etc.)
environment
Sensing System
Neural Network

Monitoring Conditions in (Proposed) Permanent Storage Facilities



Two facilities have been proposed for permanently storing waste from the Hanford Site. Low level waste would be mixed with grout and stored in an underground vault. High level waste would be vitrified and stored in an underground repository. *ANNs could be used to monitor the condition of the stored waste.*

There are many real-time and remote sensing applications on the Hanford Site including insitu monitoring of contaminants, and chemical and isotope identification. Many of these applications require an inexpensive, compact, and automated system for identifying and monitoring the object of interest (e.g., chemical, isotope). Such a system could be constructed with a sensor array and an automated pattern recognition system (such as a neural network). In hazardous environments, these systems have a distinct advantage over traditional sampling and laboratory analysis methods since an environment can be monitored without risk to human operators. The complexity of the data collected by large sensor arrays makes analysis with conventional methods difficult. ANNs, which are relatively easy to train for analyzing complex data, are likely to be a better choice for sensor data analysis.

InSitu Waste and Contaminant Identification

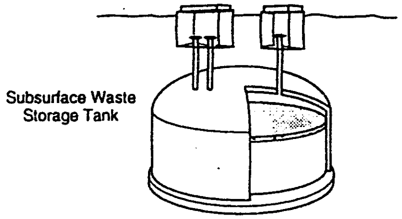
There are an estimated 1700 waste sites distributed around the 1400 square kilometers (560 square miles) of south-eastern Washington that comprise the Hanford Site. This waste includes nuclear waste (e.g., fission products), toxic chemical waste (e.g., carbon tetrachloride, ferro-cyanide, nitrates, etc.), and mixed waste (combined radioactive and chemical waste). *An ANN coupled to a sensor array (artificial nose) could be used in the contaminant identification.*

^{90}Sr ^{137}Cs CCl_4 $Fe(CN)_6^{-4}$

^{60}Co ^{239}Pu $C_6H_{12}O$ NO_3^-

Waste Storage Tank Monitoring

High level waste is currently stored in 149 single-shell and 28 double-shell, subsurface tanks. *An ANN could be used to monitor the conditions of the stored waste and alert operators of abnormalities.*



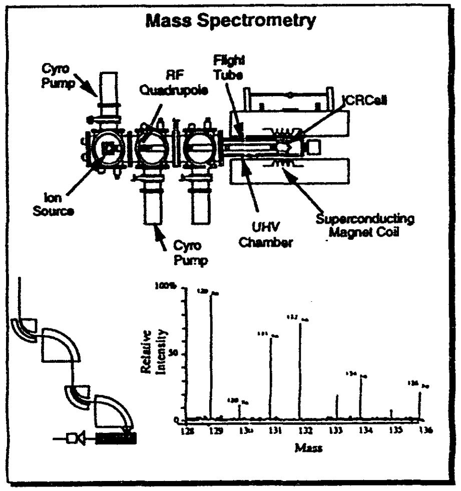
Potential Applications

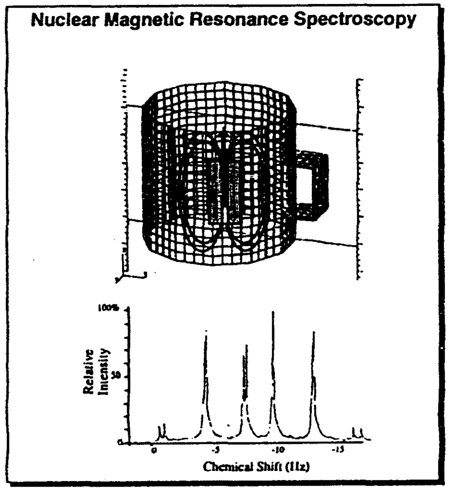
- ***Artificial Noses:***
  Artificial noses (chemical sensor array coupled to an automated chemical identification system) will be used to examine and identify chemical waste samples and contamination on the Hanford Site. Artificial noses that incorporate ANNs have been used in applications including monitoring food and beverage odors, automated flavor control, analyzing fuel mixtures, and quantifying individual components in gas mixtures.
- ***Sensor Calibration and Validation:***
  The development of new sensors requires that a methodology for sensor calibration and validation be established. ANNs have been used in spectral peak verification and will be considered for both the calibration and validation of new sensors, particularly for new complex sensors that may perform better than established calibration and validation methods.
- ***Hot-Cell Monitoring:***
  Another project on the Hanford Site involves the deployment of a multi-instrument array of fiber optic sensors, radiation sensors, and ultrasonic devices into hot cells.

Approaches

- ***Backpropagation, Feed-Forward Networks***
- ***Kohonen's Self-Organizing Networks***
- ***Hamming Networks***
- ***Boltzmann Machines***
- ***Hopfield Networks***

Spectral Analysis


Mass Spectrometry


Nuclear Magnetic Resonance Spectroscopy

The EMSL will be equipped with advanced mass spectrometers, ion cyclotron resonance mass spectrometers, and several high-field and ultrahigh-field nuclear magnetic resonance spectrometers. These instruments will be used in the analysis of large macromolecules, such as enzymes, to be used in environmental remediation.

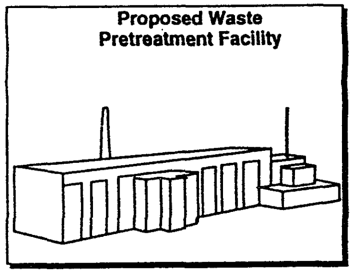
Potential Applications

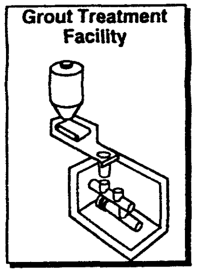
- ***Automated Identification of Spectral Data:***
  The chemical composition of a sample is determined from its ː ˉctral signature. ANNs have been successfully used to classify spectra from various moaalities including infrared spectroscopy, mass spectrometry, and NMR spectroscopy
- ***Interpretation of Important Features in Spectral Data:***
  A specific problem in this area is locating the spectral peaks of the lowest molecular-weight monoisotope in a mass spectrum of a large organic molecule. Potentially, an ANN could be trained to look at the distribution around the various peaks in the mass spectrum and infer the location of the lowest molecular-weight monoisotope.

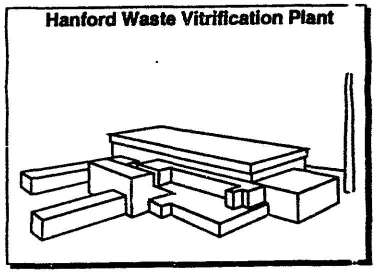
Approach

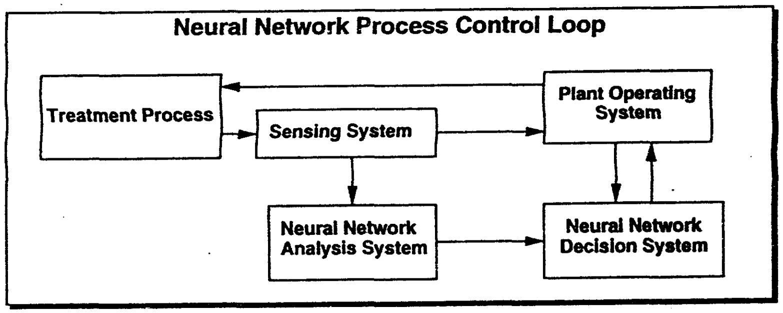
- ***Backpropagation, Feed-Forward Network:***
  Often, the backpropagation algorithm is used to train a feed-forward ANN for this application. A training set of labeled spectra are generated and presented to the training algorithm, which iteratively fixes the synaptic weights in the ANN.
- ***Optimal Linear Associative Memory:***
  The optimal linear associative memory has been used to classify components in gamma ray spectra. The ANN stores pattern data in a more compact form than the database that results in a more efficient search. Also, when it is implemented in a true parallel distributed processing system, the inherent parallelism of the ANN provides for a very rapid search.

Process Control


Proposed Waste Pretreatment Facility


Grout Treatment Facility


Hanford Waste Vitrification Plant


Neural Network Process Control Loop
Treatment Process
Sensing System
Plant Operating System
Neural Network Analysis System
Neural Network Decision System

The cleanup of Hanford will require that many controls be maintained over complex chemical processes. It would be difficult for human operators to closely monitor all key process parameters for a sophisticated chemical process in real-time. It is more effective to use automated systems in the process control and use human operators in a supervisory capacity.

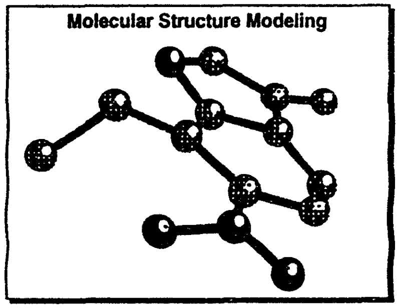
Potential Applications

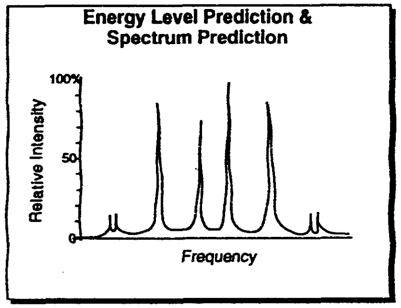
- ***Process Control:***
  ANNs allow continuous, high-level monitoring of all process sensors and can function as adaptive controllers. In many systems, performance degrades over time due to deterioration of the system components. To compensate, operational parameters are dynamically adjusted to optimize system performance. An ANN can be used to monitor the process, make decisions about system operation, and adjust the appropriate controls to keep the process operating with optimal efficiency and safety. An advantage ANNs have over more traditional adaptive controllers is that the ANN can be continuously updated with new information by using a dynamic learning approach.

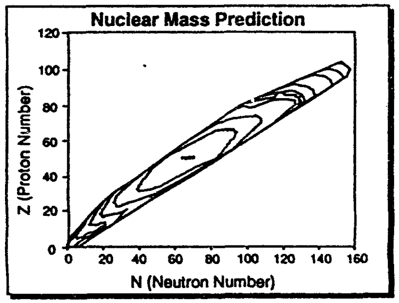
Approach

- ***Backpropagation, Feed-Forward Network:***
  The backpropagation algorithm is commonly used to train ANNs in process control with the training set composed of historical data about the process. ANNs have been used in various process control applications including process fault diagnosis and temperature.

Theoretical Modeling


Molecular Structure Modeling


Energy Level Prediction & Spectrum Prediction


Nuclear Mass Prediction

A fundamental understanding of the processes that occur in the complex environment of hazardous waste is necessary in the development of efficient and cost effective systems for environmental remediation. By simulating molecular structures and dynamics, one can gain insight into these processes.

Potential Applications

- ***Prediction of Mass Excess in the Nucleus of Isotopes***
- ***Pattern Recognition of Molecular Structures***
- ***Modeling Chemical Systems***
- ***Determination of Protein and Other Molecular Structures***
- ***Prediction of Spectral Data***
- ***Prediction of Energy Levels***

Approaches

- ***Boltzmann Machines and Hopfield Networks:***

Several theoretical models in molecular science involve search and optimization. For example, molecular structures can be determined by optimizing a set of structural parameters for a set of physical constraints. While generally producing suboptimal results, Boltzmann machines and Hopfield Networks have been used to generate approximate solutions in relatively short computation times when compared with more rigorous optimization techniques.

- ***Backpropagation, Feed-Forward Network:***

Backpropagation trained ANNs have been used to predict the secondary and tertiary structures of proteins.

Data Compression

The amount of data generated in the planned EMSL is likely to be overwhelming; therefore, construction of novel systems capable of compressing large quantities of data is necessary.

Potential Applications

- ***Principal Component Analysis (PCA):***

In an analytical system currently in use, photon counting of fluorescent molecules is performed. This procedure produces a two-dimensional histogram or image of the fluorescing surface. The size of the generated image is 1024 by 1024 pixels at 16 bits per pixel, which is equivalent to 2 million bytes of data. Larger images will be generated by systems in the EMSL. An examination of the structures in the image shows that only a small amount of information would be lost if a large amount of data compression was performed. A recognized approach in data compression of this form is to tile the image into subimages and then compress each individual subimage by using Principal Component Analysis.

Approach

- ***Backpropagation, Feed-Forward Network:***

ANNs have been trained with the backpropagation algorithm to perform efficient PCA data compression in real-time.

Conclusions

This poster has identified several real-time data processing applications in the planned EMSL that can potentially benefit from ANNs. These applications include sensor data acquisition and analysis, spectral analysis, process control, data compression, and theoretical modeling.

We are currently working on prototype evaluation to determine if ANNs are appropriate for the aforementioned applications. This involves development of software ANN simulators and exploration of the capabilities and limitations of ANNs to these applications. If ANN solutions are judged appropriate after the evaluation, then a dedicated ANN hardware system will be considered.

Comparison of Nonrecurrent Associative Memory Models Using Image Data

Paul E. Keller[1], Mariappan S. Nadar,[2] Bobby Hunt,[2] Eric VonColln,[3] Anupam Goyal[2]

[1]Pacific Northwest Laboratory, Molecular Science Research Center
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov

[2]University of Arizona, Electrical and Computer Engineering Department,
Neural Analysis and Imaging Lab, Tucson, AZ 87521
phone: (602) 621-6178 internet: nadar@nail.ece.arizona.edu

[3]NRaD, Code 7304, San Diego, CA 92152-5000

Three optimal associative memories are compared by their ability to recall image data. The first model is the classical optimal linear associative memory. The second model is an optimal nonlinear associative memory that uses a second order polynomial mapping of the input data. The third model is composed of a nonlinear transformation in the input pattern's spectral domain followed by the classical optimal linear associative memory. Computer simulations with images are used to evaluate the performance of the three models.

1. Optimal Linear Associative Memory

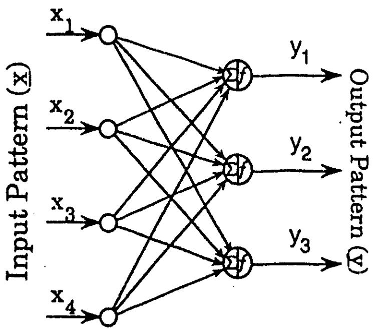
- Teuvo Kohonen, "Correlation matrix memories," *IEEE Transactions on Computers*, vol. C-21, p. 353, 1972.
- The optimal linear associative memory (OLAM) is based on a simple matrix associative memory model. It is an improvement over the original matrix memory in that it projects an input pattern onto a set of orthogonal vectors where each orthogonal vector represents a unique pattern (exemplar). With linear activation functions, the training is a straight forward matrix orthogonalization process where each pattern from the training set is made to project onto a separate, unique orthogonal axis in the output space.

Weight Specification

1. Form input and target matrices. Arrange input patterns as columns in an nxp dimensional matrix X and target patterns as columns in an mxp dimensional matrix T.

2. Generate inverse of the input pattern matrix X. Since X is generally not a square matrix, a pseudo-inverse technique is used to generate $X^\dagger$.

3. Form synaptic weight matrix.
$W_{OLAM} = TX^\dagger$ where † indicates pseudo-inverse



2. Optimal Non-Linear Associative Memory

- T. Poggio, "On optimal nonlinear associative recall," *Biological Cybernetics*, vol. 19, pp. 201-209.
- The Optimal Non-Linear Associative Memory is a modification to the OLAM that uses a polynomial mapping of the input space. The input space is transformed by:

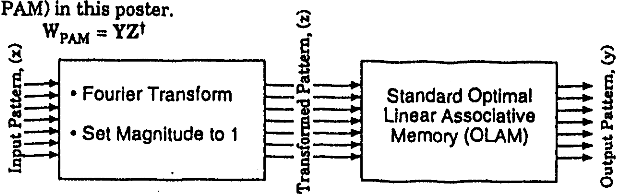
$$Z(X) = P_0 + P_1(X) + P_2(X,X) + P_3(X,X,X) + \dots + P_d(X,X,\dots)$$

where P_0, P_1, P_2, ..., P_d, are polynomial coefficients. The synaptic weights are specified in a manner similar to the OLAM.
$W_{SAM} = YZ^\dagger$
In this poster, a second order polynomial associative memory (SAM) is used.

$$Z(X) = P_0 + P_1(X) + P_2(X,X)$$

3. Optimal Linear Associative Memory with Non-Linear Preprocessing in the Spectral Domain

- B. Hunt, M. Nadar, P. Keller, E. VonColln, A. Goyal, "Synthesis of a Nonrecurrent Associative Memory Model Based on a Nonlinear Transformation in the Spectral Domain," *IEEE Transactions on Neural Networks*, vol. 4, pp. 873-878, 1993.
- This model is composed of a nonlinear transformation in the spectral domain followed by the standard optimal linear associative memory. In this poster, the input space is transformed by taking the Fourier transform of the input and setting its magnitude to unity.
$Z = \Phi(X)$ where Φ is phase portion of the Fourier domain
The prime motivation for performing this transformation is the importance of the phase of the Fourier domain representation of images in the recovery of images. This Fourier domain phase-only transformation of the input space combined with the OLAM is denoted as a phase associative memory (PAM) in this poster.
$W_{PAM} = YZ^\dagger$


Input Pattern, (x)
• Fourier Transform
• Set Magnitude to 1
Transformed Pattern, (z)
Standard Optimal Linear Associative Memory (OLAM)
Output Pattern, (y)

Results

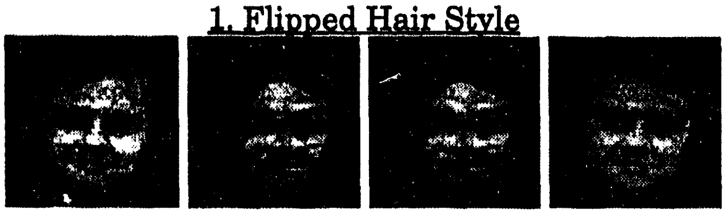
Input and Output Signal-To-Noise Ratios for Distorted Images (in dBs)

Distortion	Input	OLAM	SAM	PAM
1. Flipped Hair Style	4.54	6.89	7.99	15.75
2. Left Half of Face	2.62	4.65	5.33	13.51
3. No Face	6.23	7.66	11.00	15.89
4. Sunglasses	5.18	12.01	12.15	12.99

Input and Output Signal-To-Noise Ratios for Noisy Images (in dBs)

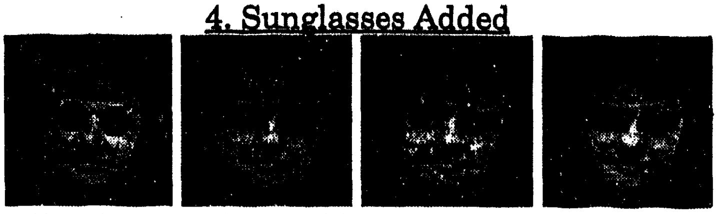
Gaussian Noise	Input	OLAM	SAM	PAM
SNR = 20	20.0	35.65	36.63	22.14
6. SNR = 10	10.0	25.65	26.64	16.59
7. SNR = 0	0.0	15.65	16.68	12.08
8. SNR = -10	-10.0	5.65	6.98	1.22
9. SNR = -20	-20.0	-4.35	-3.20	---
SNR = -30	-30.0	-14.35	-24.11	---

Images Recalled From Three Associative Memories



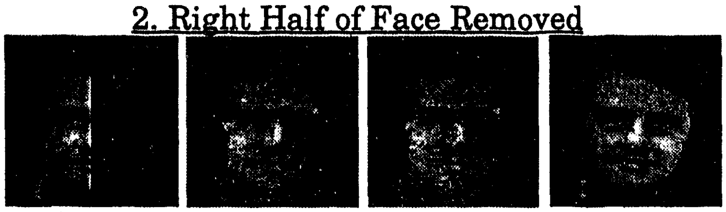
1. Flipped Hair Style

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



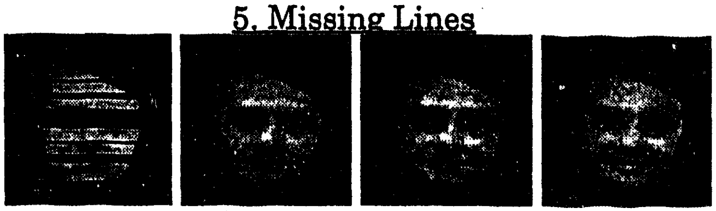
4. Sunglasses Added

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



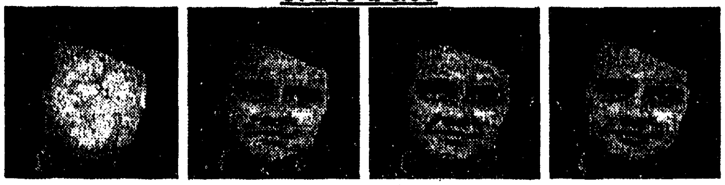
2. Right Half of Face Removed

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



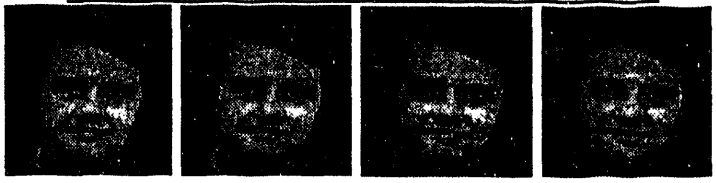
5. Missing Lines

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



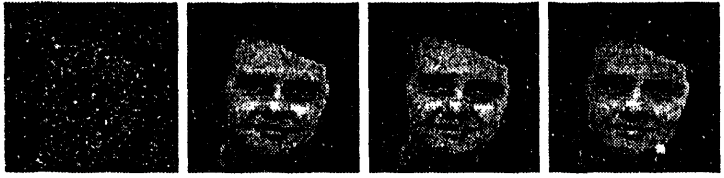
3. No Face

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



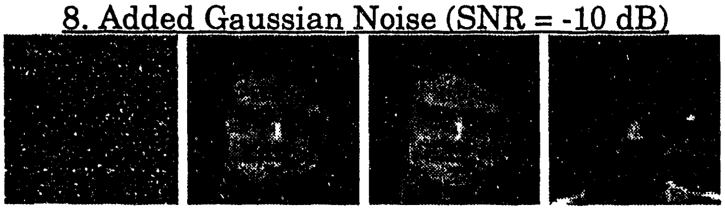
6. Added Gaussian Noise (SNR = 10 dB)

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



7. Added Gaussian Noise (SNR = 0 dB)

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled



8. Added Gaussian Noise (SNR = -10 dB)

(a) Input Image (b) OLAM Recalled (c) SAM Recalled (d) PAM Recalled

Conclusion

- For Distortions of the Images (e.g., partial removal or modification), the OLAM with spectral domain preprocessing (PAM) performed best of the three techniques tested.

- For Noisy Images, the Second Order Associative Memory (SAM) performed best, though only slightly better than the standard OLAM.

- The spectral domain processing used in the PAM (phase of the Fourier transform) produces a system sensitive to high spatial frequencies. These frequencies contain much of the unique detail of an image. Large scale modification of the images (distortions) is a low spatial frequency effect. Adding noise, affects the high spatial frequencies more than the low spatial frequencies. This explains the performance of the PAM.

NEURAL
AND
IMAGE
LABORATORY
The University of Arizona

An Optical Neural Network Implemented with Fixed, Planar Holographic Interconnects

Paul E. Keller[1] and Arthur F. Gmitro[2]
[1]Pacific Northwest Laboratory, Molecular Science Research Center, Computing and Information Science
K1-87, P.O. Box 999, Richland, WA 99352
phone: (509) 375-2254 fax: (509) 375-6631 internet: pe_keller@pnl.gov
and
[2]University of Arizona, Optical Sciences Center and Department of Radiology
Tucson, AZ 87521
phone: (602) 626-4720 fax: (602) 694-2521 internet: gmitro@zen.radiology.arizona.edu

A key element of most neural network systems is the massive number of weighted interconnections used to tie relatively simple processing elements (neurons) together in a useful architecture. The inherent parallelism and interconnection capability of optics make it a likely candidate for the implementation of the neural network interconnection process. While there are several optical technologies currently under investigation, this poster presents an optical system that combines fixed planar holographic interconnects and opto-electronic neurons.

An analysis of this architecture shows a potential interconnection capacity of 45 million synaptic connections for a planar hologram. The dynamic range of each synaptic connection is about 38:1 (approximate precision of 5 bits). Higher interconnection densities can be achieved by accepting a lower dynamic range. For opto-electronic neurons employing laser diodes, processing rates of 45 to 720 trillion connections per second can potentially be achieved.

An experimental optical system employing binary amplitude holograms is also presented in this poster. This experimental system encodes a Hopfield auto-associative memory and demonstrates the ability of this optical architecture to implement the structure of the neural network.

Objectives

(1) Investigate Feasibility of Constructing an Optically Implemented Neural Network that uses Fixed Planar Holographic Interconnects

(2) Evaluate Planar Holographic Interconnection Technology

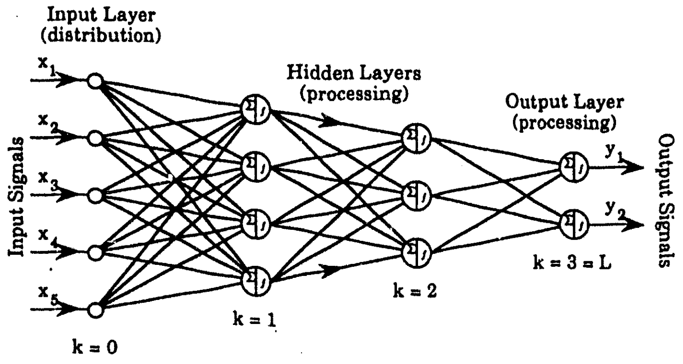
(3) Develop New Computer Generated Holograms Techniques for the Proposed System

(4) Evaluate Necessary Opto-Electronic Components

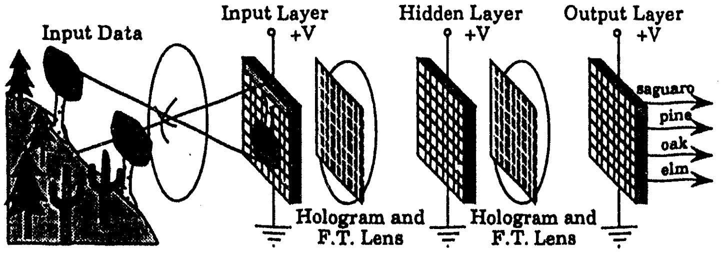
(5) Demonstrate a Prototype System

(6) Determine Potential Capabilities and Limitations of This Approach

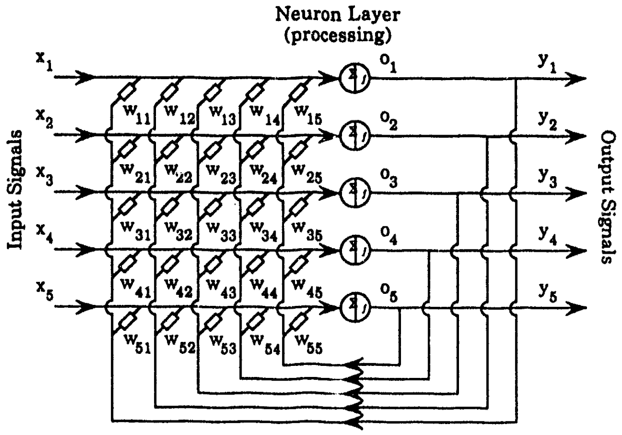
Feedforward Neural Network



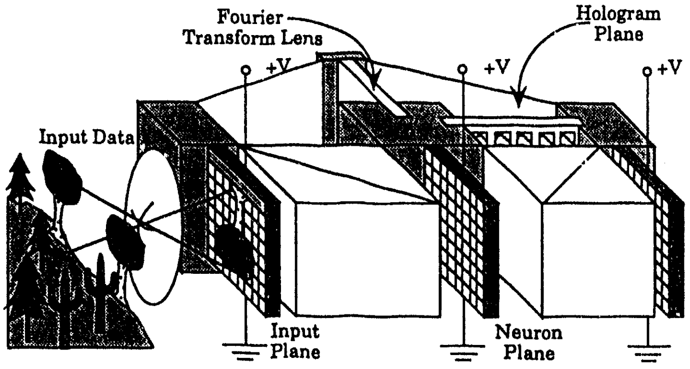
Optical Implementation

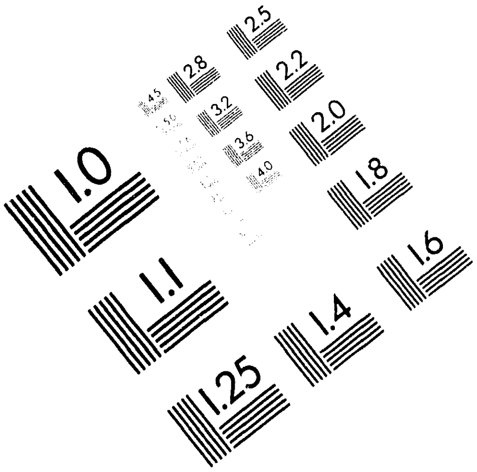


Feedback Neural Network



Optical Implementation



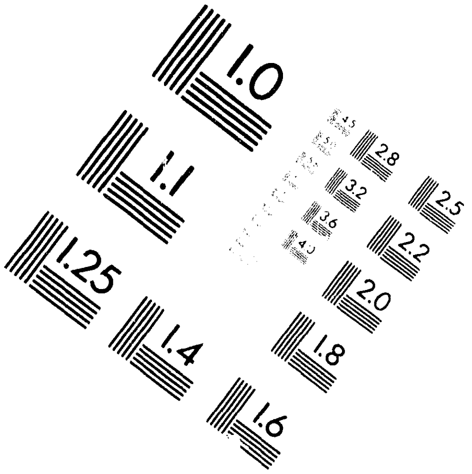


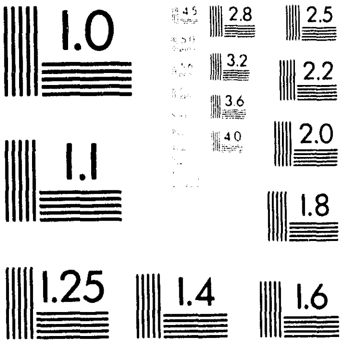
AIIM

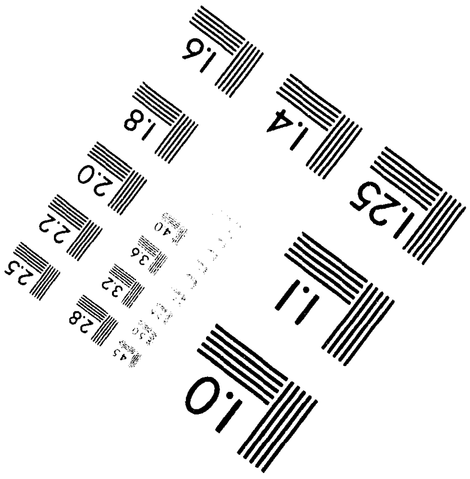
Association for Information and Image Management

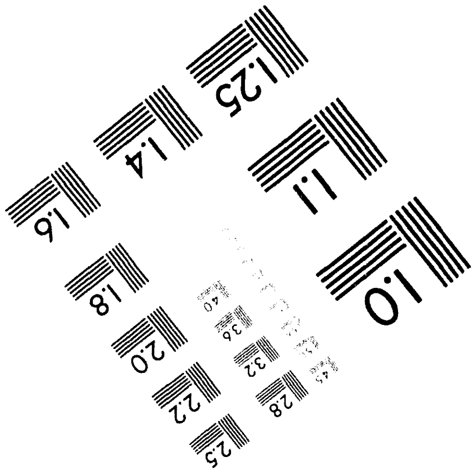
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202







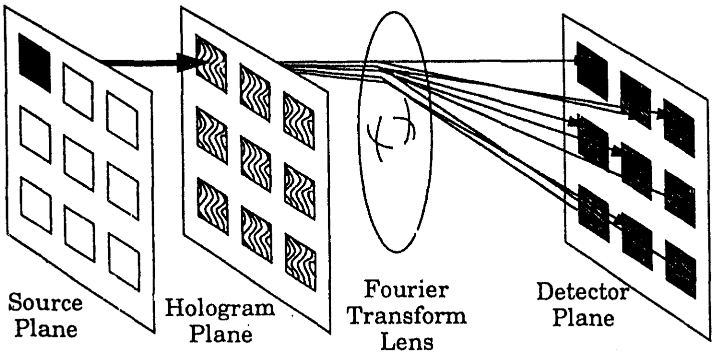


MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

Optical Interconnections

⇒ Why use Optical Interconnections?

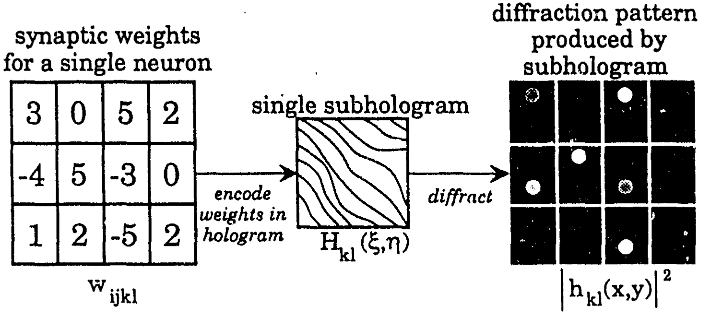
no interaction between optical beams traversing the same space



- Each neuron output has its own hologram

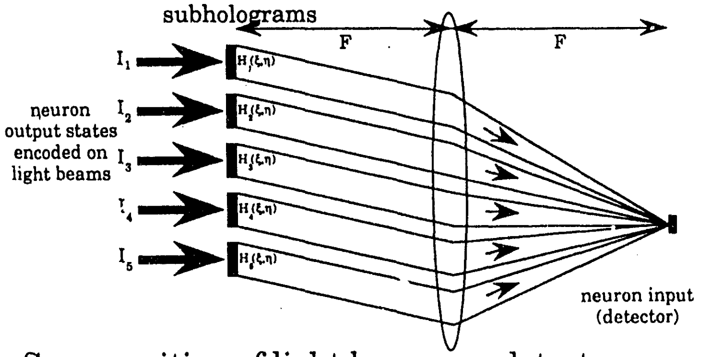
This prototype illustrates how holographic interconnects can be used to connect one layer of optical sources to another plane of optical detectors. An individual neuron emits a beam of light that illuminates a single subhologram. The Fraunhofer diffraction pattern produced by the subhologram is composed of a set of light beams that connect this output signal to each detector. The intensity of each beam arriving at the detector plane represents the strength of the synaptic connection between two neurons.

- Connection (synaptic) weights are stored in the diffraction pattern of the hologram


synaptic weights for a single neuron
single subhologram
diffraction pattern produced by subhologram
encode weights in hologram
diffract
$H_{kl}(\xi,\eta)$
w_{ijkl}
$|h_{kl}(x,y)|^2$

3	0	5	2
-4	5	-3	0
1	2	-5	2

- Fourier transform holograms map angles to detector locations

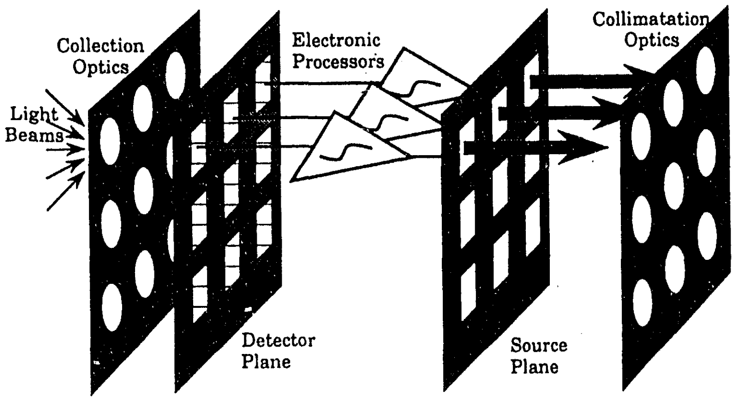


- Superposition of light beams on detector (incoherent assumption)

Opto-Electronic Neurons

⇒ Why use Opto-Electronic Neurons?

nonlinear operations are inefficient in optical materials

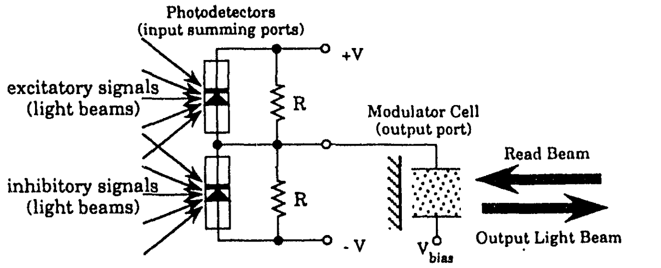


- The synaptic weight of each connection is represented by the power falling on a detector cell

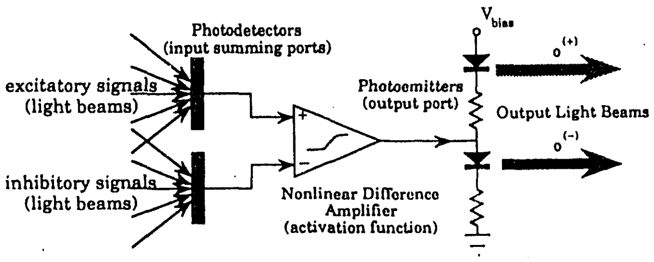
This figure illustrates a prototype opto-electronic neuron plane. Each neuron consists of a pair of input detectors (which convert optical signals to electronic), a non-linear electronic amplifier that implements a neuron activation function, and an output light source or spatial light modulator. The intensity of the output beam encodes the state of the neuron.

- Bipolar weights are encoded by spatial separation of positive weights and negative weights falling on the detector cell

Detector/Modulator Opto-Electronic Neuron

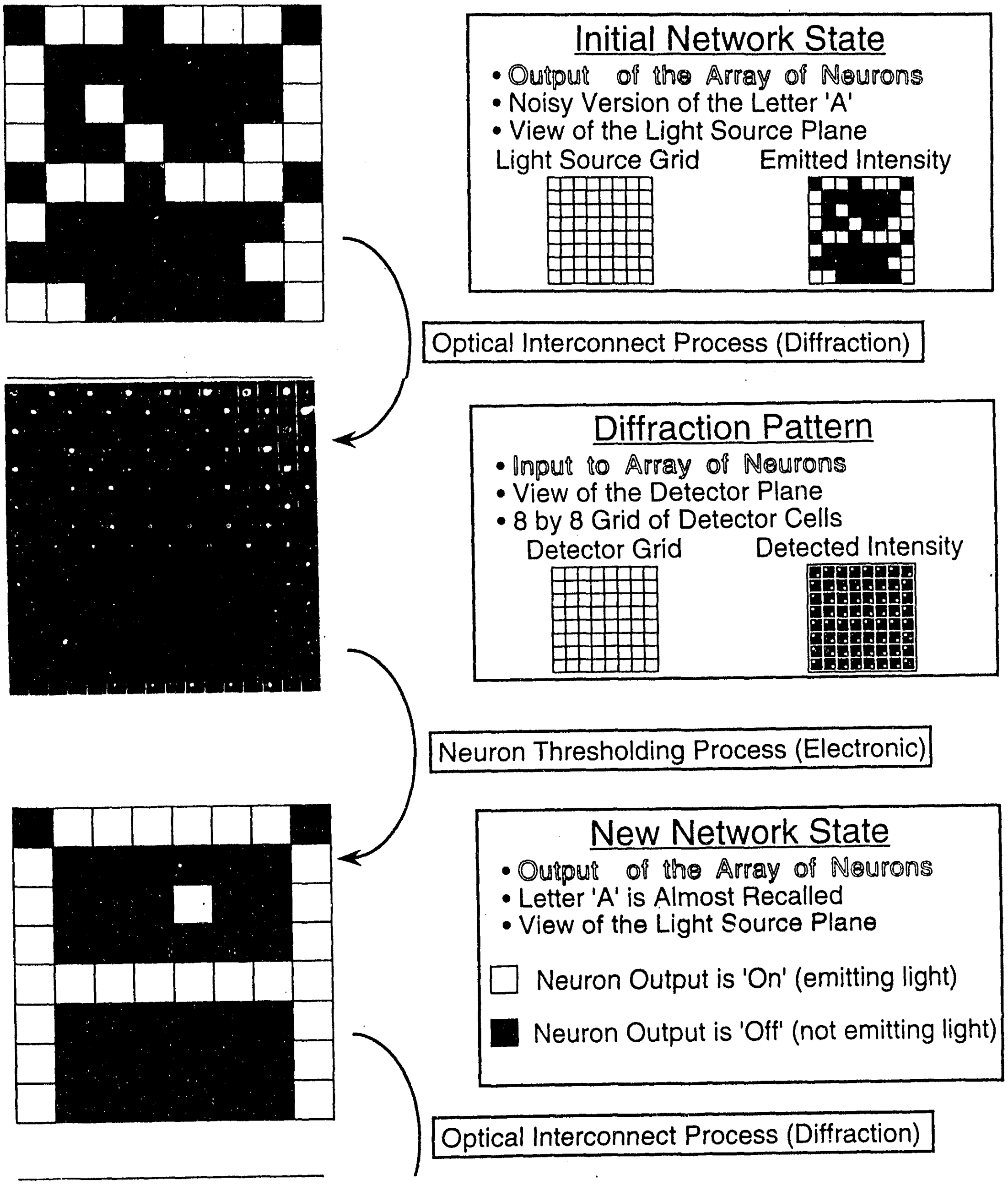

Photodetectors (input summing ports)
+V
excitatory signals (light beams)
R
Modulator Cell (output port)
Read Beam
inhibitory signals (light beams)
R
-V
V_{bias}
Output Light Beam

Detector/Emitter Opto-Electronic Neuron


Photodetectors (input summing ports)
V_{bias}
$o^{(+)}$
excitatory signals (light beams)
Photoemitters (output port)
Output Light Beams
$o^{(-)}$
+
−
inhibitory signals (light beams)
Nonlinear Difference Amplifier (activation function)

Operation of the Experimental System

The following sequence of patterns was generated with the optical system when it was configured as a Hopfield autoassociative memory. The Hopfield network is a single layer feedback system where the synaptic weights are specified by using an outer-product rule. This network tries to associate each pattern presented it with a pattern stored in the network. This example, illustrates how a corrupted or noisy version of the letter A is recalled by the network. The letter A is one of the stored patterns.



Initial Network State

- Output of the Array of Neurons
- Noisy Version of the Letter 'A'
- View of the Light Source Plane

Light Source Grid   Emitted Intensity

Optical Interconnect Process (Diffraction)

Diffraction Pattern

- Input to Array of Neurons
- View of the Detector Plane
- 8 by 8 Grid of Detector Cells

Detector Grid   Detected Intensity

Neuron Thresholding Process (Electronic)

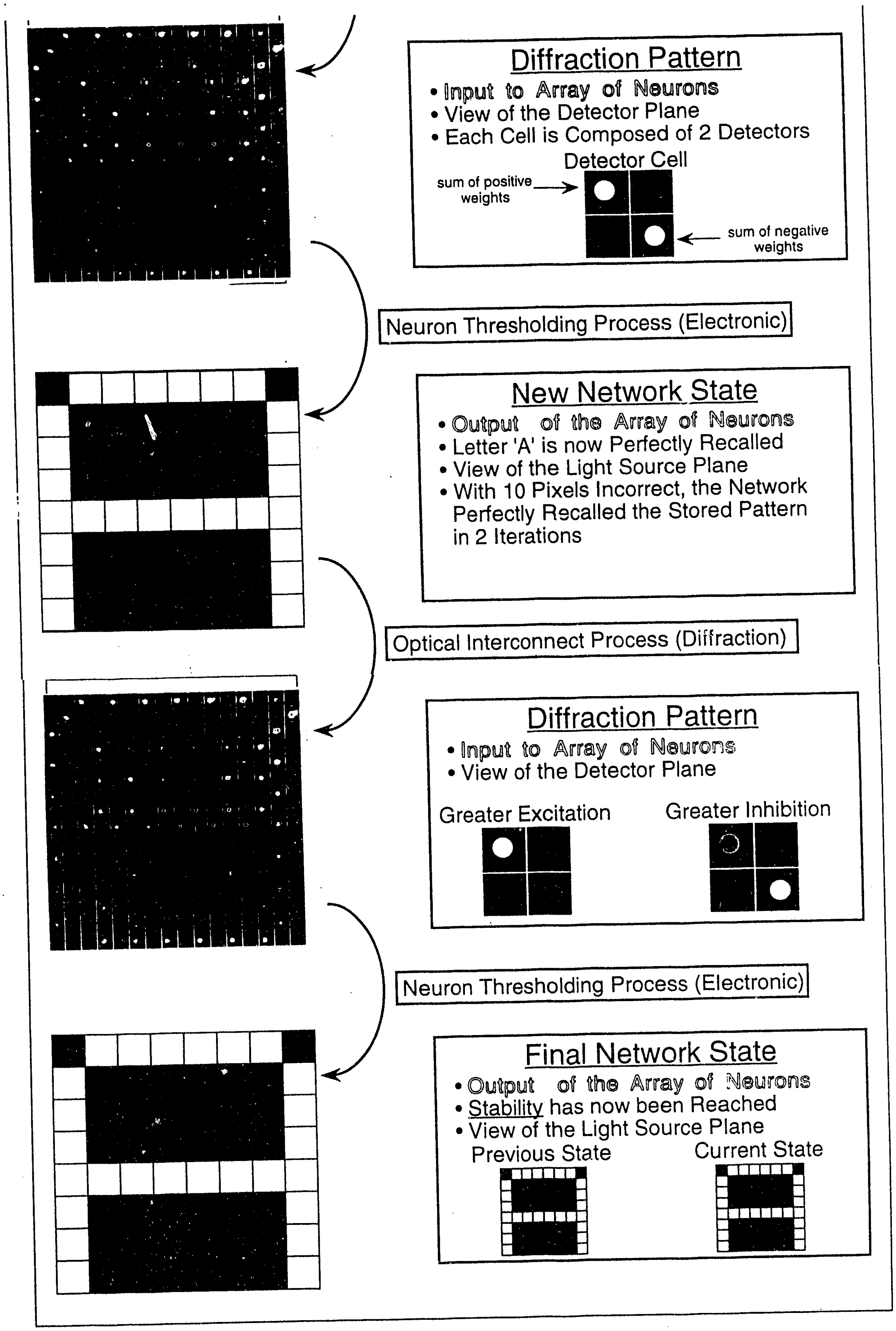
New Network State

- Output of the Array of Neurons
- Letter 'A' is Almost Recalled
- View of the Light Source Plane

□ Neuron Output is 'On' (emitting light)

■ Neuron Output is 'Off' (not emitting light)

Optical Interconnect Process (Diffraction)



Diffraction Pattern

- Input to Array of Neurons
- View of the Detector Plane
- Each Cell is Composed of 2 Detectors

Neuron Thresholding Process (Electronic)

New Network State

- Output of the Array of Neurons
- Letter 'A' is now Perfectly Recalled
- View of the Light Source Plane
- With 10 Pixels Incorrect, the Network Perfectly Recalled the Stored Pattern in 2 Iterations

Optical Interconnect Process (Diffraction)

Diffraction Pattern

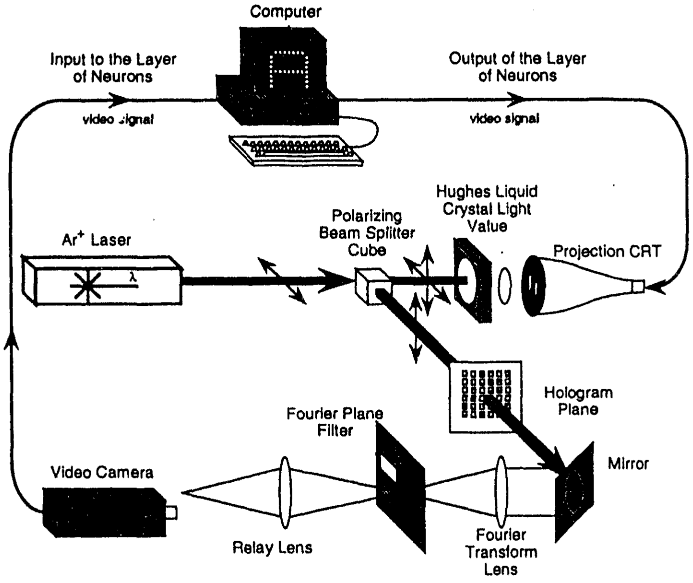
- Input to Array of Neurons
- View of the Detector Plane

Neuron Thresholding Process (Electronic)

Final Network State

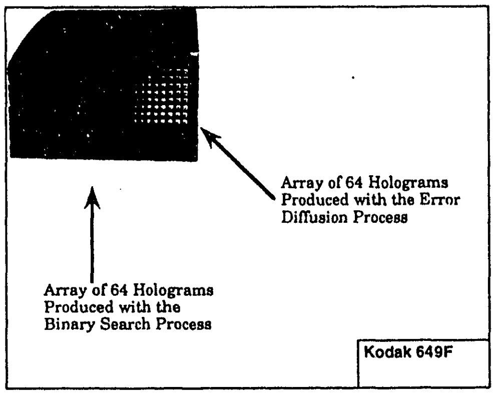
- Output of the Array of Neurons
- Stability has now been Reached
- View of the Light Source Plane

Experimental Opto-Electronic Neural Network



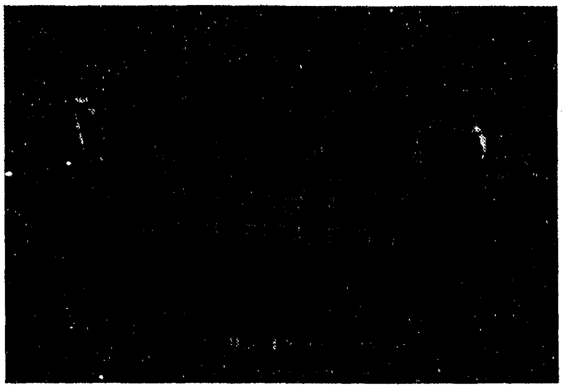
This figure illustrates the complete opto-electronic neural network configured for the feedback architecture. While this experimental system is not practical for the implementation of real-world applications, it is useful in proving the technology and for evaluating the performance of holographic optical interconnects.

Interconnection Holograms


Array of 64 Holograms Produced with the Error Diffusion Process
Array of 64 Holograms Produced with the Binary Search Process
Kodak 649F

This holographic plate contains the synaptic weights for two complete Hopfield associative memory neural networks. This plate was is used in the experimental opto-electronic neural network. There are two arrays of holograms. One array of 64 holograms for each neural network. Each hologram is 1 mm square in size and encodes the weights interconnecting a single neuron to all other neurons in the network. These holograms are computer generated. The array on the left was generated by using a random binary search process, and the array on the right was generated by using the error diffusion process. This holographic plate was produced during a two step photographic process. The size of each hologram pixel is 4μm by 4μm.

Photographs of the Experimental System



Side view of the experimental opto-electronic neural network. The high intensity projection CRT and liquid crystal light valve are pictured on the right. The video camera is pictured on the left. The argon ion laser is pictured in the background.

Top view of the experimental opto-electronic neural network. The high intensity projection CRT and liquid crystal light valve are pictured on the right. The video camera is pictured on the lower left. The argon ion laser is pictured on the top.

Potential Capabilities of Planar Holographic Optical Neural Networks

Neurons per Layer	Synapses per Layer	Approx. Precision	Coding Device	Processing Rate (CPS)	Power per Neuron
26,912	7.2x10^{8}	3.9 - 4.2 bits	VCSEL	7.2x10^{14}	180 - 220 μW
			SLM	7.2x10^{11}	
13,448	1.8x10^{8}	4.3 - 4.9 bits	VCSEL	1.8x10^{14}	90 - 110 μW
			SLM	1.8x10^{11}	
6728	4.5x10^{7}	4.9 - 5.2 bits	VCSEL	4.5x10^{13}	57 - 130 μW
			SLM	4.5x10^{10}	

CPS = Connections per Second

VCSEL = Vertical Cavity Surface Emitting Laser Diode

- switching rates exceeding 1 MHz
- speed limited by detectivity
- power dissipation limitations

SLM = Spatial Light Modulator

- switching rates of 1 - 5 kHz

Conclusions

- Small Scaled (64 neuron) Neural Network was Implemented and Demonstrated with an Optical System
- Performance of Optical System was Similar to Computer Simulations of the System
- Advantages of This Approach:
  - True Parallel Processing
  - Very Fast
    - Processing Rates Upto 720 Trillion Connections Per Second
  - Can Implement Large Systems
    - Upper Limit of 6,800-27,000 Neurons Per Layer
- Disadvantages of This Approach:
  - No On-line Learning
  - Limited Dynamic Range
    - 40:1 (equivalent digital precision of 5.5 bits)
  - Immature technology (currently not practical)
- Since training must be done off-line and generation of the interconnection hologram is computationally expensive, optical neural networks constructed with fixed, planar holograms are best suited to situations where very high speed processing is required, the trained system is to be mass produced, and infrequent or no relearning is anticipated.

Additional Work

P. Keller and A. Gmitro, "Operational Parameters of an Opto-Electronic Neural Network Employing Fixed Planar Holographic Interconnects", *Proceedings of the World Congress on Neural Networks*, Volume 4, pp. 799-802, (International Neural Network Society, Washington, DC, 1993).

P. Keller and A. Gmitro, "Computer-generated holograms for neural networks: analysis of on-axis and off-axis diffraction geometries," *Applied Optics* , Volume 32, pp. 1304-1310 (1993).

P. Keller and A. Gmitro, "Design and analysis of fixed planar holographic interconnects for optically implemented neural networks," *Applied Optics* , Volume 31, pp. 5517-5526 (1992).

P. Keller and A. Gmitro, "Computer generated planar holograms for optical neural network implementations," *Annual Meeting of the Optical Society of America*, Albuquerque, NM, 25 September 1992.

P. Keller and A. Gmitro, "Design and Demonstration of an Opto-Electronic Neural Network using Fixed Planar Holographic Interconnects," in *Topical Meeting on Optical Computing, Technical Digest Series*, Volume 6, (Optical Society of America, Washington, DC, 1991) pp. 80-83.

A. Gmitro, P. Keller, G. Gindi, "Statistical performance of outer-product associative memory models," *Applied Optics* , Volume 28, pp.1940-1948 (1989).

A. Gmitro and P. Keller, "Space-Variant Interconnects Via Multifaceted Planar Holograms, " *Annual Meeting of the Optical Society of America*, Santa Clara, CA, 4 November 1988.

Author Index

Doc I

DATE FILMED

8/12/94

END