

PORT DOCI

AD-A277 507



1a. REPORT SECURITY CLASSIFICATION

Unclassified

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

DTIC
ELECTE
MAR 30 1994
S F D

Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

#69

6a. NAME OF PERFORMING ORGANIZATION

Institute for Brain and Neural Systems

6b. OFFICE SYMBOL (if applicable)

7a. NAME OF MONITORING ORGANIZATION

Personnel and Training Research Programs
Office of Naval Research (Code 1142PT)

6c. ADDRESS (City, State, and ZIP Code)

Brown University
Providence, Rhode Island 02912

7b. ADDRESS (City, State, and ZIP Code)

800 North Quincy Street
Arlington, VA 22217-5000

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (if applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

N00014-91-J-1316

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

Proceedings of the NIPS'93 Postconference Workshop on Methods for Combining Neural Networks

12. PERSONAL AUTHOR(S)

Michael P. Perrone

13a. TYPE OF REPORT

Technical Report

13b. TIME COVERED

FROM TO

14. DATE OF REPORT (Year, Month, Day)

March 15, 1994

15. PAGE COUNT

51(2-sided)

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
05	08	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Averaging, Ensemble Methods, Committees

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report is the proceedings from the NIPS'93 postconference workshop entitled "Pulling It All Together: Methods for Combining Neural Networks." The goal of the workshop was to examine theoretical aspects and heuristic methods for combining existing neural network algorithms to generate systems which have improved performance properties.

This report includes an outline and summary of the workshop as well as the slides used in each participant's presentation.

DTIC QUALITY INSPECTED 3

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT

☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

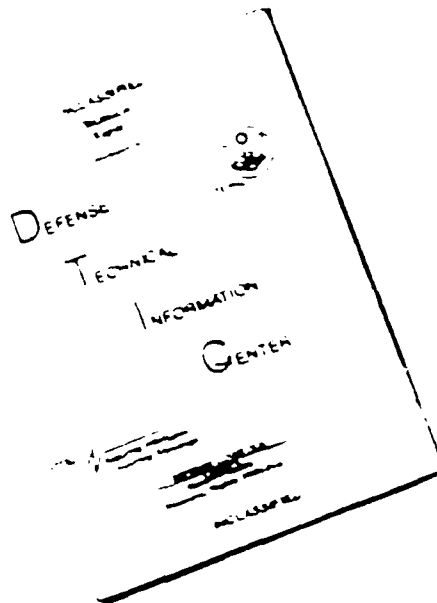
Dr. Joel Davis

22b. TELEPHONE (Include Area Code)

(703) 696-4744

22c. OFFICE SYMBOL

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST
QUALITY AVAILABLE. THE COPY
FURNISHED TO DTIC CONTAINED
A SIGNIFICANT NUMBER OF
PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

Pulling It All Together: Methods for Combining Neural Networks*

Michael P. Perrone
Institute for Brain and Neural Systems
Brown University
Providence, RI
mper@cs.brown.edu

[This is a brief summary of the workshop. The overhead slides used by the speakers following this summary.]

The past several years have seen a tremendous growth in the complexity of the recognition, estimation and control tasks expected of neural networks. In solving these tasks, one is faced with a large variety of learning algorithms and a vast selection of possible network architectures. After all the training, how does one know which is the best network? This decision is further complicated by the fact that standard techniques can be severely limited by problems such as overfitting, data sparsity and local optima. The usual solution to these problems is a winner-take-all cross-validation model selection. However, recent experimental and theoretical work indicates that we can improve performance by considering methods for combining neural networks.

This workshop examined current neural network optimization methods based on combining estimates and task decomposition, including Boosting, Competing Experts, Ensemble Averaging, Metropolis considerations, Stacked Generalization and Stacked Regression. The issues covered included Bayesian error orthogonality, task decomposition, network selection techniques, overfitting, data sparsity and local optima. Highlights of each talk are given below. To obtain the workshop proceedings, please contact the author or Norma Carrara (norma.carrara@brown.edu) and ask for IBNS ONR technical report #68.

M. Perrone (Brown University, "Averaging Methods: Theoretical Issues and Real World Examples") presented weighted averaging schemes [7], discussed their theoretical foundation [6], and showed that averaging can improve performance whenever the cost function is (positive or negative) convex which includes Mean Square Error, a general class of L_p -norm cost functions, Maximum Likelihood Estimation, Maximum Entropy, Maximum Mutual Information, the Kullback-Leibler Information (Cross Entropy), Penalized Maximum Likelihood Estimation and Smoothing Splines [6]. Averaging was shown to improve performance on the NIST OCR data, a human face recognition task and a time series prediction task [9].

J. Friedman (Stanford, "A New Approach to Multiple Outputs Using Stacking") presented a detailed

*To appear in "Neural Information Processing Systems 6," S. J. Hanson, J. D. Cowan and C. L. Giles (eds.), Morgan Kaufmann, 1994.

analysis of a method for averaging estimators, and noted simulations showed that averaging with a positivity constraint was better than cross-validation estimator selection [1]. S. Nowlan (Synaptics, "Competing Experts") emphasized the distinctions between static and dynamic algorithms and between averaged and stacked algorithms, and presented results of the mixture of experts algorithm [3] on a vowel recognition task and a hand tracking task.

H. Drucker (MIT, "Boosting Compared to Other Ensemble Methods") reviewed the boosting algorithm [2] and showed how it can improve performance for OCR data.

J. Moody (OGI, "Predicting the U.S. Index of Industrial Production") showed that neural networks make better predictions for the US IP index than standard models [4] and that averaging these estimates improves prediction performance further.

W. Buntine (NASA Ames Research Center, "Averaging and Probabilistic Networks: Automating the Process") discussed placing combination techniques within the Bayesian framework.

D. Wolpert (Santa Fe Institute, "Inferring a Function vs. Inferring an Inference Algorithm") argued that theory can not, in general, identify the optimal network, so one must make assumptions in order to improve performance.

H. Thodberg (Danish Meat Research Institute, "Error Bars on Predictions from Deviations among Committee Members (within Bayesian Backprop)") raised the provocative (and contentious) point that Bayesian arguments support averaging while Occam's Razor (seemingly) does not.

S. Basilev (Purdue University, "Merits of Combining Neural Networks: Potential Benefits and Risks") emphasized the importance of dealing with collinearity when using averaging methods.

References

- [1] Leo Breiman, "Stacked regression," Technical Report TR-367, Department of Statistics, University of California, Berkeley, August 1992.
- [2] Harris Drucker, Robert Schapire, and Patrick Simard, "Boosting performance in neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, [To appear].
- [3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, 3(2), 1991.
- [4] U. Levin, T. Leen, and J. Moody, "Fast pruning using principal components," In Steven J. Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, 1994.
- [5] M. P. Perrone, "Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Optimization," PhD thesis, Brown University, Institute for Brain and Neural Systems, Dr. Leon N. Cooper, Thesis Supervisor, May 1993.
- [6] M. P. Perrone, "General averaging results for convex optimization," In *Proceedings of the 1993 Connectionist Models Summer School*, pages 364-371, Hillsdale, NJ 1993, Erlbaum Associates.
- [7] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble method for neural networks," In *Artificial Neural Networks for Speech and Vision* (Chapman-Hall, 1993), Chapter 10.

Pulling It All Together: Methods for Combining Neural Networks
NIPS*93 Postconference Workshop
December 4, 1993

Intended Audience

Those interested in optimization algorithms for improving neural network performance.

Goal

Stimulate discussion on methods for combining neural networks

Organizer

Michael P. Perrone
Institute For Brain and Neural Systems, Brown University
Prometheus Inc., Newport RI
Email: mpp@cns.brown.edu

Pulling It All Together - NIPS'93 Workshop

7:30-7:35	Opening Remarks
7:35-7:55	M. Perrowe, (Brown University) "Averaging Methods: Theoretical Issues and Real World Example"
7:55-8:15	J. Friedman, (Stanford) "A New Approach to Multiple Outputs Using Stacking"
8:15-8:35	S. Nowlan, (Synaptics) "Competing Experts"
8:35-8:55	H. Drucker, (AT&T) "Boosting, Compared to Other Ensemble Methods"
8:55-9:30+	Discussion
9:30-4:30	FREE TIME
Cancelled	C. Scofield, (Nestor Inc.) (Applications)
4:30-4:50	J. Moody (OGI) "Forecasting the U.S. Index of Industrial Production"
4:50-5:10	W. Buntine, (NASA Ames Research Center) "Averaging and Probabilistic Networks: Automating the Process"
5:10-5:30	D. Wolpert, (Santa Fe Institute) "Inferring a Function vs. Inferring an Inference Algorithm"
5:30-5:50	H. Thodberg, (Danish Meat Research Institute) "Error Bars on Predictions from Deviations among Committee Members (within Bayesian Backprop)"
5:50-6:10	S. Hashem, (Purdue University) "Merits of Combining Neural Networks: Potential Benefits and Risks"
6:10-6:30+	Discussion & Closing Remarks
7:00	Workshop Wrap-Up (common to all sessions)

94-09622



25

Topics

- Boosting Algorithm
- Competing Experts Algorithm
- Ensemble Averaging Algorithm
- Stacked Generalization Algorithm
- Stacked Regression Algorithm
- Network Selection Techniques
- Bayesian Methods
- Cross Validation
- Incorporating A Priori Knowledge
- Error Orthogonality
- Task Decomposition
- Overcoming Data Sparsity
- Using Local Optima
- Overfitting

Accession For	NTIS GRA&I	DTIC TAB	Unannounced	Justification	By	Distribution /	Availability	Dist	41
---------------	------------	----------	-------------	---------------	----	----------------	--------------	------	----

Averaging Methods: Theoretical Issues and Real World Examples

OUTLINE

- Problem of Multiple Neural Networks
- Ensemble Method
 - Basic Algorithm
 - Generalized Algorithm
 - Theoretical Basis
- Experimental Results
 - NIST OCR Database
 - Turk & Pentland Human Face Database
 - Sunspot Database
- Extensions
 - Convexity
 - Other Cost Functions
 - Variance Reduction

Michael P. Perrone

Institute for Brain and Neural Systems

Brown University

Email: mpp@cns.brown.edu

This research was supported by the Office of Naval Research, the Army Research Office and the National Science Foundation.

PROBLEM

- Many learning algorithms
 - Many possible architectures
 - Many local minima
- Many
networks
- ⇒ **disagreeing**

Naive estimate:

- Choose the best on an independent test set
- **We can do better!**

Combine networks:

- Bayesian BP (Buntine & Weigend 92)
- Hierarchical NNs (Ersoy & Hong 90)
- Hybrid NNs (Cooper 91; Scofield, et al 87; Reilly 88, 87)
- Local Experts (Jacobs, et al 91)
- Neural Trees (Perrone 92a, 92b; Sankar 90)
- Stacked Generalization (Wolpert 90)
- Synergy (Lincoln & Skrzypek 90)
- (Etc.)

⇒ Ensemble Method

- Simple
- Improves estimate
- Theoretical basis

Basic Ensemble Method

Given:

$(x, y) \sim \mathcal{P}$
 $\mathcal{D} \equiv$ Training set
 $\mathcal{CV} \equiv$ Cross-Validation set
 $\mathcal{T} \equiv$ Testing set
 Network set: $\{f_i(x)\}_{i=1}^N$ trained on $\mathcal{D}$

Find:

$$f(x) = E_{\mathcal{P}}[y|x]$$

Ensemble Regression Function:

$$f_{\text{ensemble}}(x) \equiv \frac{1}{N} \sum_{i=1}^N f_i(x)$$

Misfit:

$$m_i(x) \equiv f(x) - f_i(x)$$

Average Individual MSE:

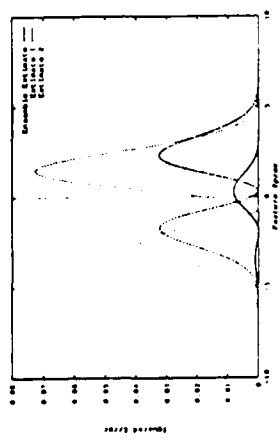
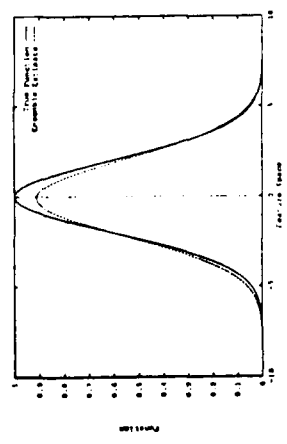
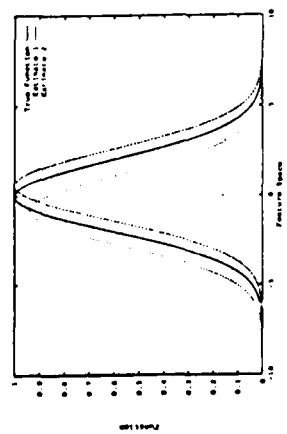
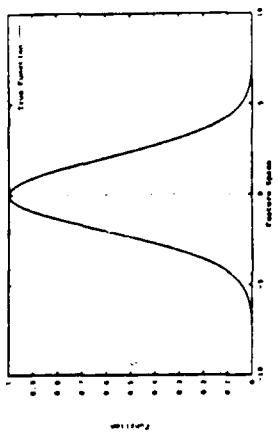
$$\overline{\text{MSE}}_{\text{individual}} = \frac{1}{N} \sum_{i=1}^N E_{\text{cv}}[m_i^2]$$

Uncorrelated, Zero Mean $m_i(x)$'s:

$$\text{MSE}_{\text{Ensemble}} = \frac{1}{N} \overline{\text{MSE}}_{\text{individual}}$$

INTUITIVE EXAMPLE

Removing the Independence Assumption



Cauchy Inequality

$$\left(\sum_{i=1}^n x_i y_i\right)^2 \leq \left(\sum_{i=1}^n x_i^2\right) \left(\sum_{i=1}^n y_i^2\right)$$

$$\left(\sum_{i=1}^n x_i\right)^2 \leq n \sum_{i=1}^n x_i^2$$

$$\text{MSE}[\bar{f}] \leq \overline{\text{MSE}[f]}$$

$$\sum_{i=1}^n (y_i - f(x_i))^2 / g^2(x_i)$$

$$y = \int f(x; w) P(w) dw.$$

$$E_w[f(x; w)] \approx \frac{1}{n} \sum_{i=1}^n f(x; w_i)$$

$$\begin{aligned} E[(\hat{f}(x) - f(x))^2] &= E[(\hat{f} - E[\hat{f}] + E[\hat{f}] - f(x))^2] \\ &= E[(\hat{f} - E[\hat{f}])^2] + E[(E[\hat{f}] - f(x))^2] \\ &\quad + 2E[(\hat{f} - E[\hat{f}])(E[\hat{f}] - f(x))] \\ &= \text{VAR}(\hat{f}) + \text{BIAS}^2(\hat{f}) \end{aligned}$$

$$\text{VAR}(\hat{f}) \equiv E[(\hat{f} - E[\hat{f}])^2]$$

$$\text{BIAS}(\hat{f}) \equiv E[\hat{f}] - f(x)$$

$$\lim_{n \rightarrow \infty} \bar{f}(x) = \int \hat{f}(x; \beta) p(\beta) d\beta$$

Generalized Ensemble Method

Weighted Average:

$$\hat{f}(x) \equiv \sum_{i=1}^{i=N} \alpha_i f_i(x), \quad \alpha_i \in \mathcal{R}$$

$$\sum \alpha_i = 1 \Rightarrow \hat{f}(x) = f(x) + \sum_{i=1}^{i=N} \alpha_i m_i(x)$$

Covariance Matrix:

$$C_{ij} \equiv E_{CV}[m_i(x)m_j(x)]$$

Minimize:

$$MSE(\alpha) = \sum_{i,j} \alpha_i \alpha_j C_{ij}$$

Result:

$$\alpha_i^{opt} = \frac{\Sigma_j C_{ij}^{-1}}{\Sigma_k \Sigma_j C_{kj}^{-1}} \quad \text{and} \quad MSE(\alpha^{opt}) = \left[\sum_{ij} C_{ij}^{-1} \right]^{-1}$$

Uncorrelated $m_i(x)$'s:

$$\alpha_i^{opt} = \frac{\sigma_i^{-2}}{\Sigma_j \sigma_j^{-2}} \quad \text{and} \quad MSE(\alpha^{opt}) = \left[\sum_i \sigma_i^{-2} \right]^{-1}$$

Unconstrained Ensemble Method

Weighted Average:

$$\hat{f}(x) \equiv \sum_{i=1}^{i=N} \alpha_i f_i(x), \quad \alpha_i \in \mathcal{R}$$

Estimate Matrix and Measurement Vector:

$$f_{j1} \equiv f_i(x_j)$$

$$F_j \equiv f(x_j)$$

Minimize:

$$MSE(\alpha) = (f\alpha - F)^T (f\alpha - F)$$

Result:

$$\alpha = (f^T f)^{-1} f^T F$$

Infinite Data:

$$(f^T f)_{ij} \rightarrow E[f_i(x)f_j(x)]$$

$$(f^T F)_i \rightarrow E[f_i(x)f(x)]$$

Entropy $H(p) = - \sum_i p(x_i) \ln p(x_i)$

$$H(p) \geq \overline{H(p)}$$

Mutual Information

$$I(a, b) = H(a) + H(b) - H(ab)$$

$$I(\bar{p}, b) \geq \overline{I(p, b)}$$

Log-Likelihood Function

$$L(p) = \prod_i p(x_i)$$

$$\ln L(\bar{p}) \geq \overline{\ln L(p)}$$

Kullback-Leibler Information

$$K(f, g) = \int f \ln\left(\frac{f}{g}\right)$$

$$\overline{K(p, g)} \geq K(\bar{p}, g)$$

Smoothing Splines

$$S(f) = \frac{1}{n} \sum_i (\hat{f}(x_i) - f_i)^2 + \lambda \int (f'')^2 dx$$

$$S(\bar{f}) \leq \overline{S(f)}$$

Convex on $[a, b]$

$$h\left(\frac{x_1 + x_2}{2}\right) \leq \frac{h(x_1) + h(x_2)}{2}$$

Jensen's inequality

$$\Phi\left(\frac{\int_a^b f(x; \omega) g(\omega) d\omega}{\int_a^b g(\omega) d\omega}\right) \leq \frac{\int_a^b \Phi(f(x; \omega)) g(\omega) d\omega}{\int_a^b g(\omega) d\omega}$$

$$\Phi(Ef) \leq E[\Phi(f)]$$

$$\Phi(\bar{f}) \leq \overline{\Phi(f)}$$

l_p -norm Extension

$$E(\{x_i\}) = \sum_i (\alpha_i |x_i|^{p_{\alpha i}} - \beta_i |x_i|^{p_{\beta i}})$$

Experimental Results

- BP nets with 2 weight layers
- Cross-validated stopping rule
- Varied the number of hidden units
- Random initial weights
- 10 nets in each ensemble
- Figure of Merit:

$$FOM \equiv \%Correct - \%Rejected - 10(\%Error)$$

NIST OCR Data

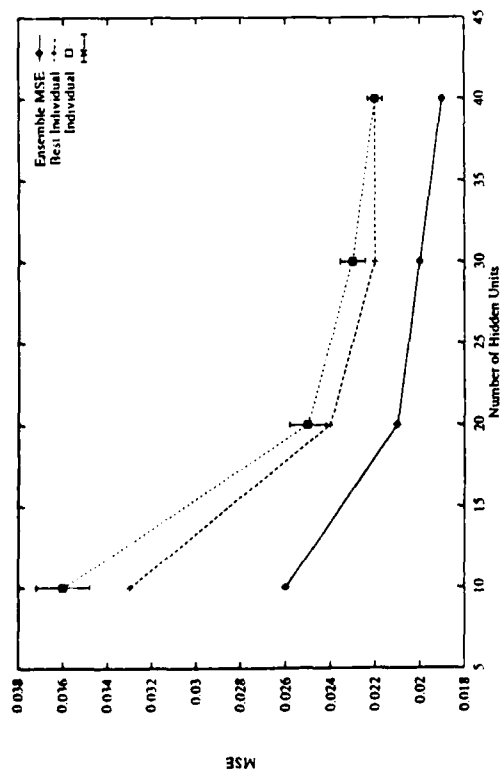
- Handwritten characters
- Hand-segmented
- Hand-labeled
- 120 dimensional feature space

(Supplied by Nestor, Inc., Providence, RI)

DATA SET	TRAINING SET	CV SET	TESTING SET	CLASSES
Numbers	13241	13241	4767	10
Uppercase	11912	11912	7078	26
Lowercase	12971	12970	6835	26

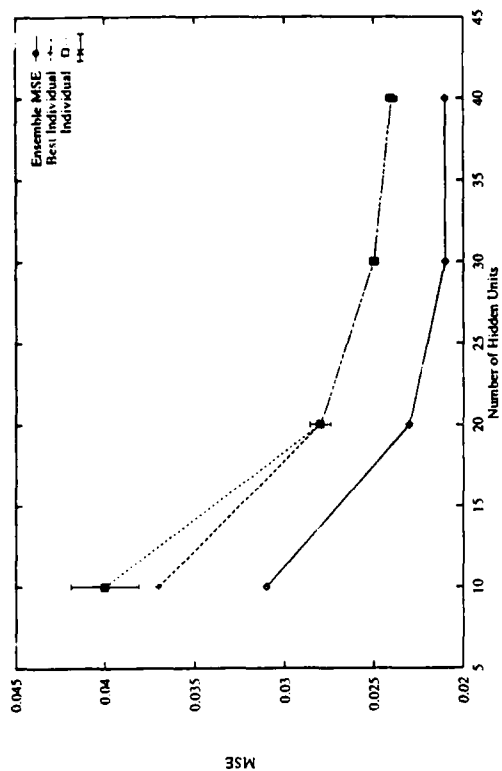
MSE vs. Hidden Units

NIST Uppercase - Independent test set



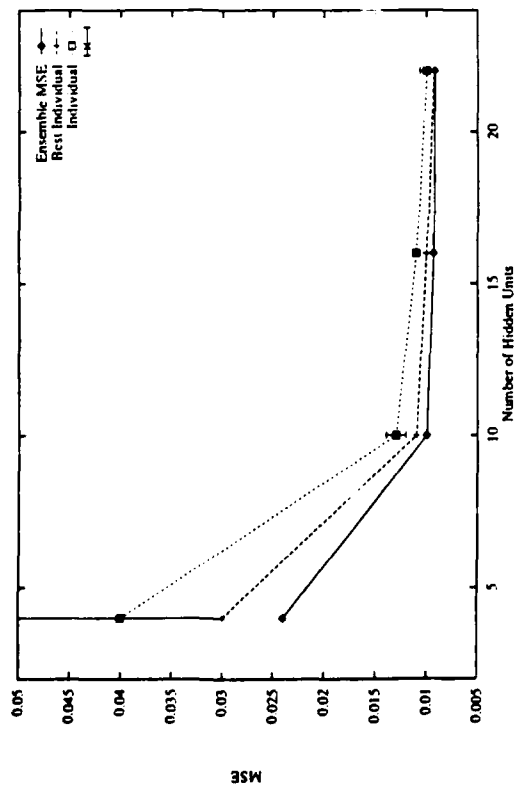
MSE vs. Hidden Units

NIST Lowercase - Independent test set



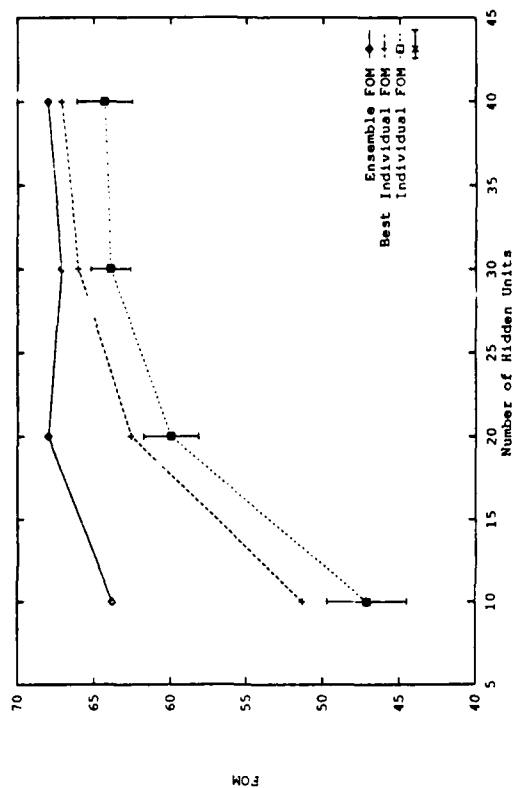
MSE vs. Hidden Units

NIST Numerals - Independent test set

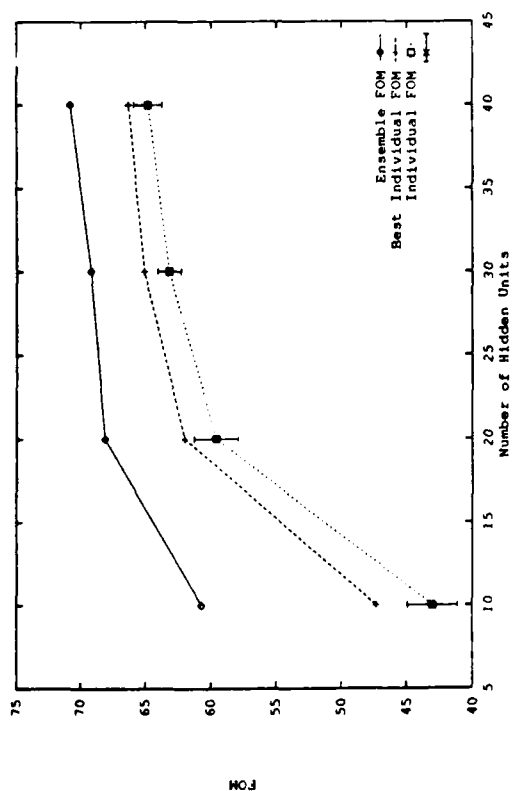


FOM vs Hidden Units

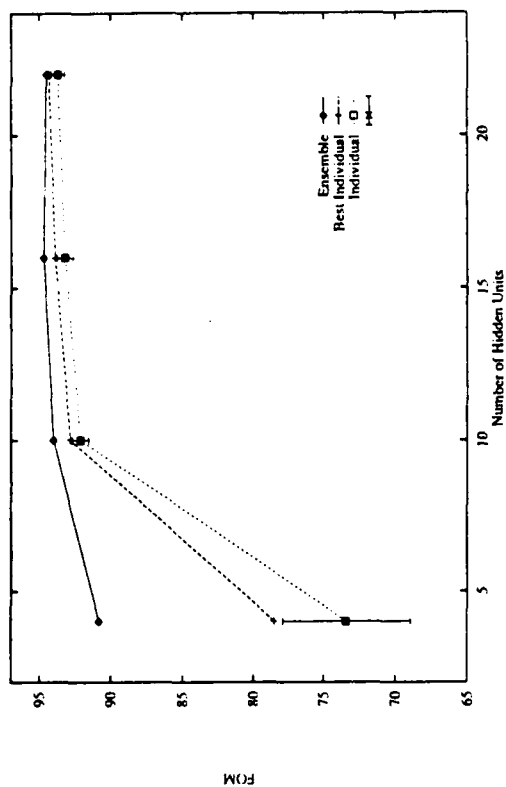
NIST Uppercase - Independent test set



FOM vs Hidden Units
NIST Lowercase - Independent test set



FOM vs Hidden Units
NIST Numbers - Independent test set



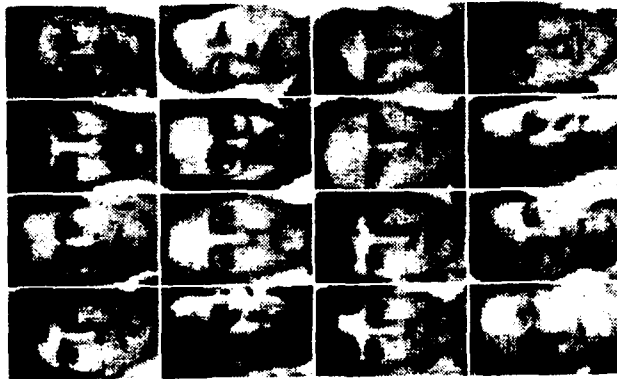
Turk and Pentland Data

- Human male face images
- "Warped" - centered and rotated
- 2294 dimensional gray-scale feature space

(Supplied by Daniel Reissfeld, Tel Aviv University)

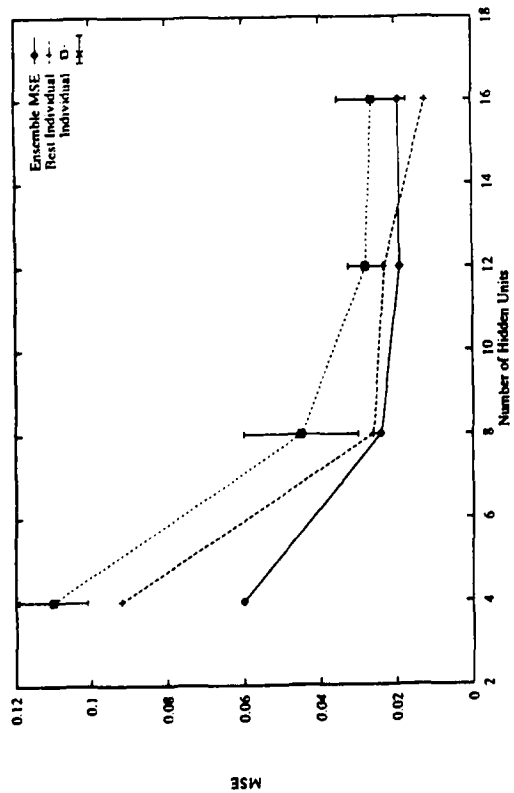
DATA SET	TRAINING SET	CV SET	TESTING SET	CLASSES
Faces	136	136	160	16

Samples of the Face Data



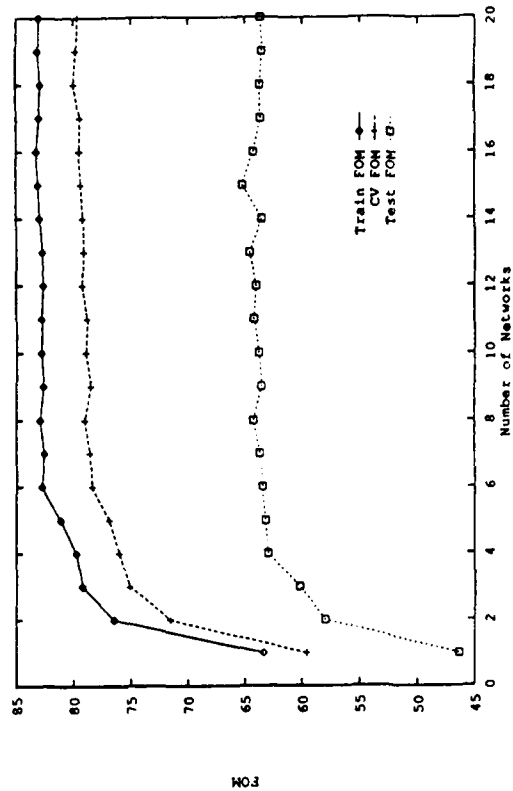
MSE vs. Hidden Units

Human Face Data - Independent test set



DEPENDENCE ON ENSEMBLE SIZE

FOM vs Number of Ensemble Nets
NIST Uppercase - 10 hidden units



Network Selection

- Multiple Data Sets

- Multiple Architectures

- Multiple Learning Rules - flexibility!

- Bootstrap Generation
 - One training set
 - Multiple trained nets

- Generalized Jackknife Generation
 - Vary the CV subset
 - Uses entire data set AND avoids over-fitting

SUMMARY

PROBLEM: Neural nets often disagree

GENERALIZED ENSEMBLE METHOD:

- Construct ensemble: **weighted average** of networks
- Choose weights by minimizing MSE (with nets fixed)

ADVANTAGES:

- Simple
- Firm theoretical basis
- Significantly improves performance
- General - Combines multiple architectures data sets and learning rules
- Extends to any convex cost optimization

A New Approach to Multiple Outputs Through Stacking

Leo Breiman
U.C. Berkeley

Jerome H. Friedman
Stanford University

Problem:

model: $\{y_m = f_m(x_1 \dots x_m) + \epsilon_m\}_{I=1}^M$
 ↑
 outputs
 ↑
 target functions
 common inputs
 ↑
 noises (maybe = 0)

training data: $\{y_{i1} \dots y_{im}; x_{i1} \dots x_{im}\}_{i=1}^N$

develop: $\{\hat{f}_m(x_1 \dots x_m)\}_{I=1}^M$ approximations

such that:

$$\{E[f_m(x) - \hat{f}_m(x)]^2\}_{I=1}^M = \text{small} \quad (\text{each})$$

def: $x = \{x_1 \dots x_m\} \in R^m$

Approach (1): separate problems

$$\hat{f}_m(x) = \underset{g(x)}{\operatorname{argmin}} \underbrace{\sum_{i=1}^N [y_{im} - g(x_i)]^2}_{\text{fit to the training data}} + \lambda_m P_m(g)$$

regularization - { strength parameter - penalty functional }

$$\hat{f}_m(x) = \sum_{k=1}^{K_m} \hat{\alpha}_{mk} B(x | \hat{x}_{mk})$$

↑
basis functions parameters

Approach (2): common estimation (usual)

$$\{\hat{f}_m(x)\}_{I=1}^M = \underset{\{g_m(x)\}_{I=1}^M}{\operatorname{argmin}} \sum_{m=1}^M \sum_{i=1}^N [y_{im} - g_m(x_i)]^2$$

$$+ \lambda P(g_1 \dots g_M)$$

single strength parameter common penalty

$$\hat{f}_m(x) = \sum_{k=1}^K \hat{\alpha}_{mk} B(x | \hat{x}_{ke})$$

common basis function set

FFNN's, PPR, CART, MARS, RBF's etc

Motivations for (2) over (1):

A. Computational

B. Statistical

if $\{f_m(\underline{x})\}_1^M$ all quite similar
then they can borrow strength
 from each other $\Rightarrow$ fewer
 (nonlinear) parameters.

Additional strength borrowing:

let $\{\beta_k = \beta(\underline{x} | \underline{y}_k)\}_1^K$ "transformed
 inputs"

$\hat{f}_m(\underline{x}) = \sum_{k=1}^K \hat{\alpha}_{mke} \beta_k$ linear model

$\{\hat{\alpha}_{mke}\} = \arg \min_{\{\alpha_{mke}\}} \sum_{m=1}^M \sum_{i=1}^N [y_{im} - \sum_{k=1}^K \alpha_{mke} \beta_{ike}]^2$

linear least squares regression

Breiman & Friedman (1993): "C&W"

Instead of $\{\hat{f}_m(\underline{x})\}_1^M$, use $\{\hat{f}_m(\underline{x})\}_1^M$

$\tilde{f}_m(\underline{x}) = \sum_{\ell=1}^M b_{m\ell} \hat{f}_\ell(\underline{x})$ linear combinations

let: $\hat{\underline{f}}(\underline{x}) = \{\hat{f}_m(\underline{x})\}_1^M \in \mathbb{R}^M$

$\tilde{\underline{f}}(\underline{x}) = \{\tilde{f}_m(\underline{x})\}_1^M \in \mathbb{R}^M$

$B = [b_{m\ell}] \in \mathbb{R}^{M \times M}$

then: $\tilde{\underline{f}}(\underline{x}) = B \hat{\underline{f}}(\underline{x})$

$\uparrow$ least squares estimate
 "shrinking" matrix

How to get B?

ideally:

$\{b_{m\ell}\}_1^M = \arg \min_{\{b_{\ell}\}_1^M} E [y_m - \sum_{\ell=1}^M b_{\ell} \hat{f}_\ell(\underline{x})]^2$
 $\uparrow$ future data (population)

optimizes each output error separately

Don't have future data
 $\Rightarrow$ cross-validation

$$\{\hat{b}_{me}\}_1^M = \underset{\{b_e\}_1^M}{\operatorname{argmin}} \sum_{i=1}^N [y_{im} - \sum_{e=1}^M b_e \hat{f}_{ei}(x_i)]^2$$

$\hat{f}_{ei} = \hat{f}_e(x)$ estimated with its training datum not in training sample.

use $\{\hat{f}_e(x)\}_1^M$ as "stacked" approx's
 for each $\hat{f}_m(x)$, $m=1, M$.

Breiman & Friedman (1993):

$$y_e = [\hat{F}_{ie}] = [\hat{f}_e(x_i)], \quad Y = [y_{ie}] \in \mathbb{R}^{N \times M}$$

$$\Phi = (Y^T Y)^{-1} Y^T \hat{F} (\hat{F}^T \hat{F})^{-1} \hat{F}^T Y \in \mathbb{R}^{M \times M}$$

$$= T D_q T^{-1} \quad (\text{eigen analysis})$$

$T \in \mathbb{R}^{M \times M}$ = eigenvectors of Φ (canonical coordinates)

$$D_q = \operatorname{diag}\{q_1^2, \dots, q_M^2\} = \text{eigen values}$$

(canonical correlations squared)

then: $\hat{B} \approx T D_b T^{-1}$ (GCV approx. to CV)

$$D_b = \operatorname{diag}\{b_1, \dots, b_M\}$$

$$b_i = \frac{(1 - K/N)(q_i^2 - K/N)}{(1 - K/N)^2 q_i^2 + (K/N)^2 (1 - q_i^2)}$$

"magic formula")

Breiman & Friedman show

$$b_i \leftarrow [b_i]_+ = \begin{cases} b_i & \text{if } b_i > 0 \\ 0 & \text{if } b_i < 0 \end{cases}$$

(positive part shrinkage)

is a BIG HELP.

(works better than real cross-validation.)

Good news: C&W "post processor" can help with approach (2) to reduce prediction error for each output.

Bad news: approach (2) does not have this property - why?

(1) Wrong criterion:

$$\text{Let } \underline{y} = \{y_m\}_1^M, \underline{f}(\underline{x}) = \{f_m(\underline{x})\}_1^M, \underline{x} \in \mathbb{R}^M$$

$$\Sigma(\underline{f}) = E[\underline{y} - \underline{f}(\underline{x})][\underline{y} - \underline{f}(\underline{x})]^T$$

residual (error) covariance matrix (population - future data)

Correct criterion:

$$\hat{\underline{f}}(\underline{x}) = \underset{\underline{g}(\underline{x})}{\operatorname{argmin}} \sum_{i=1}^N [\underline{y}_i - \underline{g}(\underline{x}_i)]^T \Sigma(\underline{f})^{-1} [\underline{y}_i - \underline{g}(\underline{x}_i)] + \lambda P(\underline{g})$$

(usual) wrong criterion $\Rightarrow \Sigma(\underline{f}) = I_M$

$\underline{f}(\underline{x})$ unknown $\Rightarrow \Sigma(\underline{f})$ unknown

$$\hat{\Sigma}(\hat{\underline{f}}) = \frac{1}{N} \sum_{i=1}^N [\underline{y}_i - \hat{\underline{f}}(\underline{x}_i)][\underline{y}_i - \hat{\underline{f}}(\underline{x}_i)]^T$$

and iterate.

this is equivalent to:

$$\hat{\underline{f}}(\underline{x}) = \underset{\underline{g}(\underline{x})}{\operatorname{argmin}} \log |\hat{\Sigma}(\underline{g})| + \lambda P(\underline{g})$$

setting $\Sigma = I_M$ gives too much influence to poorly estimated $f_m(\underline{x})$ (disaster);

(2, common penalty) $\Rightarrow$ common basis function set.

if $\{f_m(\underline{x})\}_1^M$ not very similar,

then compromise $\{B_{jk}(\underline{x})\}_{j,k=1}^K$

can be bad for some (or all) $\{\hat{f}_m(\underline{x})\}_1^M$

Especially if $\Sigma = I \Rightarrow$ poorly estimatable $\hat{f}_m(\underline{x})$ dominate basis function selection.

(3) single regularization (strength)

parameter:

• compromise value

$$\hat{\lambda} = \arg \min_{\lambda} \sum_{m=1}^M \text{MSE}[\hat{f}_m(x|\lambda)]$$

↑
estimate of future prediction error

can be bad for some (or all) $\hat{f}_m(x)$

Perfectly estimated $\hat{f}_m(x)$ dominate here $\rightarrow \infty$.

• Approach (2) with (without) C&W can make things worse (much) than separate approx's (approach (1)),

especially for $f_m(x)$ that can be approx'd well.

Desire: borrow strength in a way that insures (expected) improvement over separate approx's (approach (1)).

(A) If $\{f_m(x)\}_1^M$ are very different $\Rightarrow$ same as separate approx's.

(B) If they are highly correlated $\Rightarrow$ big improvement.

How?

Idea (1): apply C&W to separate approx's.

$$\hat{f}_m(x) = \arg \min_{g(x)} \sum_{i=1}^N [y_{im} - g(x_i)]^2 + \lambda_m f_m(g)$$

$$\Phi = (Y^T Y)^{-1} Y^T \hat{F} (\hat{F}^T \hat{F})^{-1} \hat{F}^T Y = T D_g T^{-1}$$

$$\hat{B} = T D_b T^{-1}; \quad D_b = \text{magic formula } \{g_i^2\}_1^M$$

$$K = \text{ave \# of parameters} = \frac{1}{M} \sum_{m=1}^M p_m(\text{eff})$$

$$\hat{f}_m(x) = \sum_{l=1}^M \hat{b}_{ml} \hat{f}_m(x), \quad m=1, M$$

↑
final approx's.

Our implementation: always include $\hat{f}_m(x)$'s original basis functions in $\hat{f}_m(x)$:

$$\hat{f}_m^*(x) = \sum_{k=1}^{K_m} \hat{\alpha}_{m,k} B(x | \hat{x}_{m,k}^*) + \sum_{l=1}^{L_m} \hat{\alpha}_{m,l} B(x | \hat{x}_{m,l}^*)$$

↑ original basis fun.'s ↑ added basis functions from other $\hat{f}_m(x)$ (if any)

If any $\{f_k(x)\}_{k \neq m}$ highly associated with $\hat{f}_m(x) \Rightarrow$ their basis functions may help $\hat{f}_m(x)$. Note: Some basis fun's will appear in several $\hat{f}_m^*(x)$.

Then: Apply C&W to $\{\hat{f}_m^*(x)\}_1^M$

(Idea 2 + Idea 1)

$$\hat{f}_m^*(x) = \sum_{l=1}^M \hat{b}_{m,l} \hat{f}_m^*(x)$$

Idea (2): Share basis functions in a way that does not degrade (separate) performance.

$$\hat{f}_m^*(x) = \sum_{k=1}^{K_m} \hat{\alpha}_{m,k} B(x | \hat{x}_{m,k}^*)$$

↑ fit to $\{y_{im}\}_1^N$ only

merge separate basis functions of all outputs into a common pool

$$\{\hat{f}_2(x)\}_1^K = \{B(x | \hat{x}_{m,k}^*)\}_{k=1}^{K_m} \quad m=1$$

$K = \sum_{m=1}^M K_m$

and select individual (optimal) subsets for each $\hat{f}_m(x) \Rightarrow$ separate model selection for each $\hat{f}_m(x)$.

$$\hat{f}_m^*(x) = \sum_{k=1}^{K_m} \hat{\alpha}_{m,k} B(x | \hat{x}_{m,k}^*)$$

Simulation studies: 100 replications per study.

Each replication (data set):

$$\{f_m(x)\}_1^M = \sum_{\ell=1}^M \alpha_{\ell} \pi_{\ell}(x) b_m(x) \beta_{\ell}(x) \}_1^M$$

$\{\beta_{\ell}(x)\}_1^M$ = fixed pre-specified fun's (uncorrelated)

$b_m \sim N(0, 1)$ random

Π = random permutation $[1, \dots, M]$

$$\alpha_{\ell} = \exp[-0.5(\ell/\bar{\ell})^2]$$

$\bar{\ell}$ controls correlation among $\{f_m(x)\}_1^M$

$$\{y_{im} = f_m(x_i) + \sigma \cdot \varepsilon_i\}_{i=1}^N \}_{m=1}^M$$

$$x_i \sim U^m(0, 1), \quad \varepsilon_i \sim N(0, 1)$$

σ controls noise level.

$\{\hat{f}_m(x)\}_1^M$ (randomly) different for each replication

1 contr's / among $\{f_m(x)\}_1^M \sim \text{const}$ over the 100 rep's.

(100 $\cdot$ M diff. target fun's / study)

For each replication:

$\hat{f}_m(x)$ = separate approx's in order of $E[f_m(x) - \hat{f}_m(x)]^2$.

$\tilde{f}_m(x)$ = corresponding post-processed approx's.

$$\lambda_m^2 = \frac{E[f_m(x) - \tilde{f}_m(x)]^2}{E[f_m(x) - \hat{f}_m(x)]^2} \quad \text{"improve - ment ratio."}$$

look at distribution $\{\lambda_m^2\}_1^M$ averaged over 100 rep's.

$E[\cdot]$ = ave. over 5000 indep. validation sample.

Study 1: $M=7$, $m=2$, $N=200$

$$B_1 = \sin(2\pi x_1), B_2 = \sin(4\pi x_1)$$

$$B_3 = \sin(2\pi x_2), B_4 = \sin(4\pi x_2)$$

$$B_5 = \sin(2\pi x_1) \sin(2\pi x_2)$$

$$B_6 = \sin(2\pi x_1) \sin(4\pi x_2)$$

$$B_7 = \sin(4\pi x_1) \sin(2\pi x_2)$$

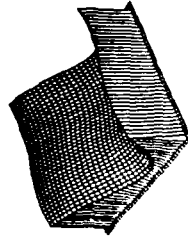
8 cases (± 100 reps each):

$$\text{ave: corr } \{f_m(x)\}_1^M = 0.8, 0.66, 0.55, 0.33$$

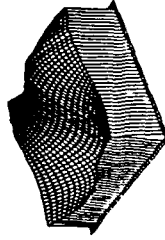
$$\sigma = 0.0, \sigma = 0.2$$

Left = trial basis fun's only

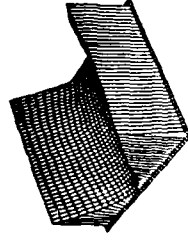
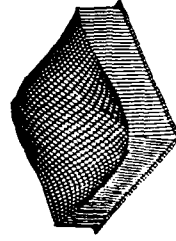
Right = trial BF's + (B&W) ("stacking")



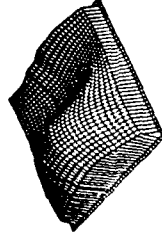
surf 4



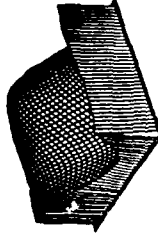
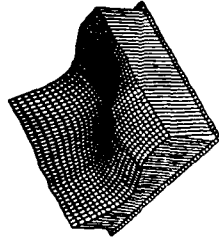
surf 7



surf 5

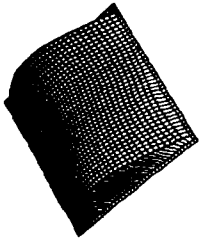


surf 6

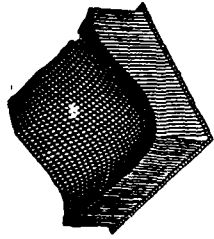


rep = 50

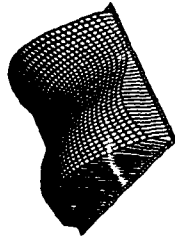
surf 2



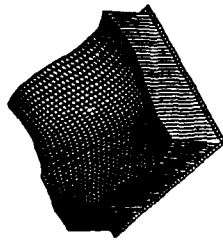
surf 3



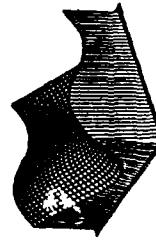
surf 5



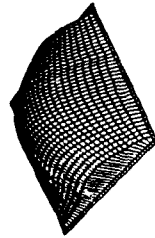
surf 4



surf 7

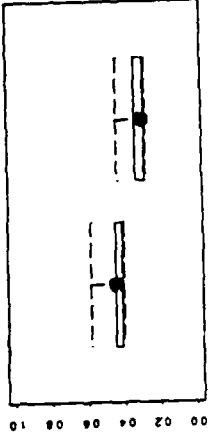


surf 6

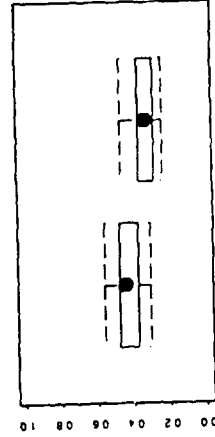


$\lambda \rho = 100$

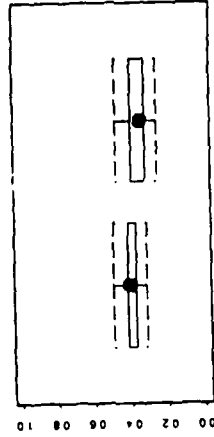
$\sigma = 0.0$



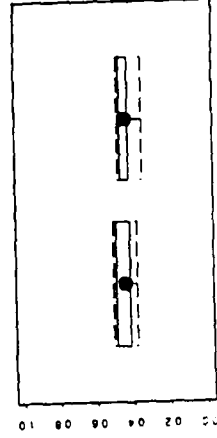
0.80



0.66

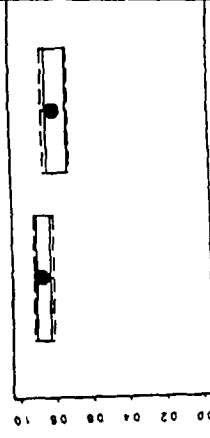
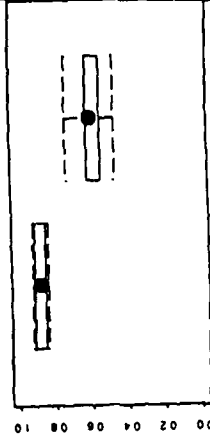
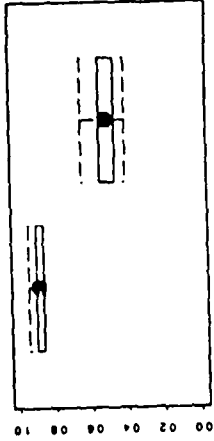
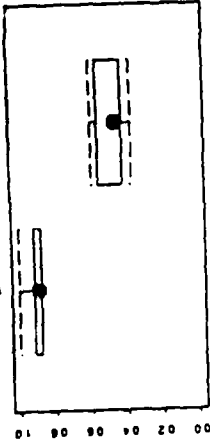


0.55



0.33

$\sigma = 0.0$



Study 2: $M=9$, $m=9$, $N=200$

$$\{B_{\ell}(x)\}_1^9 = \{\sin(5\pi x_{\ell})\}_1^9$$

independent

$$\circ \circ \{f_m(x)\} = \sum_{\ell=1}^M a_{\ell} \pi_{\ell} b \quad B_{\ell}(x)\}_1^M$$

additive

case 1: $b_{m\ell} \sim N(0, 1)$, $\bar{x} = 2$

$$\text{ave! corr } \{f_m(x)\}_1^M | = 0.65$$

$$\sigma = 0.0, 0.15$$

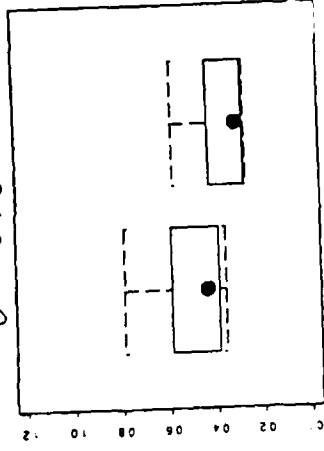
$$\text{Case 2: } b_{m\ell} = \delta_{m\ell} = \begin{cases} 1 & m=\ell \\ 0 & m \neq \ell \end{cases}$$

$$\Rightarrow \{f_m(x)\}_1^M = \text{independent}$$

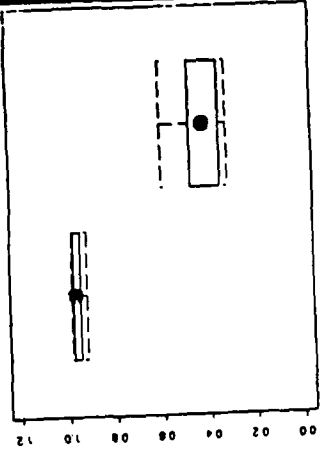
$$\Rightarrow \text{separate } \{f_m(x)\}_1^M = \text{optimal}$$

Question: How much worse are the post processed $\{\hat{f}_m(x)\}_1^M$?

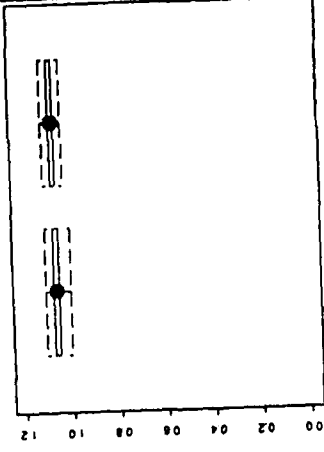
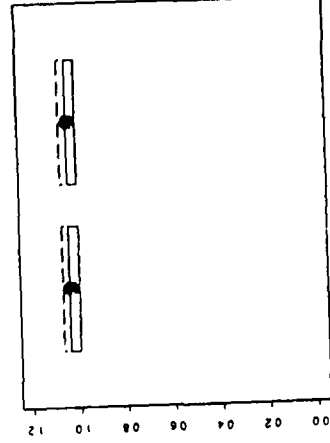
$U = 0.0$



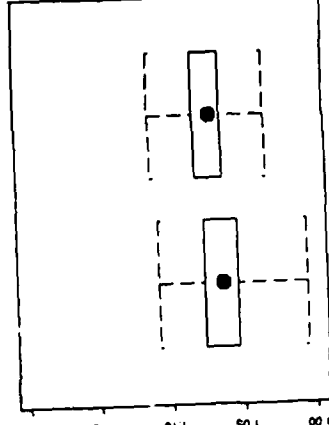
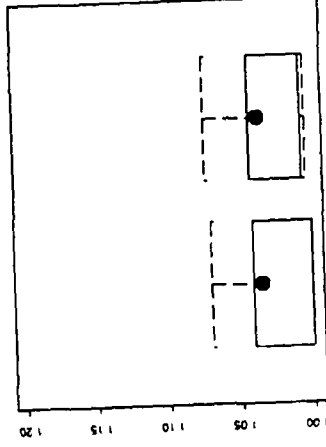
$\sigma = 0.15$



0.65



0.0



0.0

Study 3: $M=9$, $m=2$, $N=200$

$$\{B_x(x) = e^{-\frac{1}{2} |x - \underline{x}_x|^2 / \sigma_x^2}\}_I^9$$

$$x \sim U^2(0, 4), \quad \sigma_x \sim U(0.7, 1.3)$$

$$\{\underline{x}_x\}_I^9 = (1, 1), \dots, (3, 3)$$

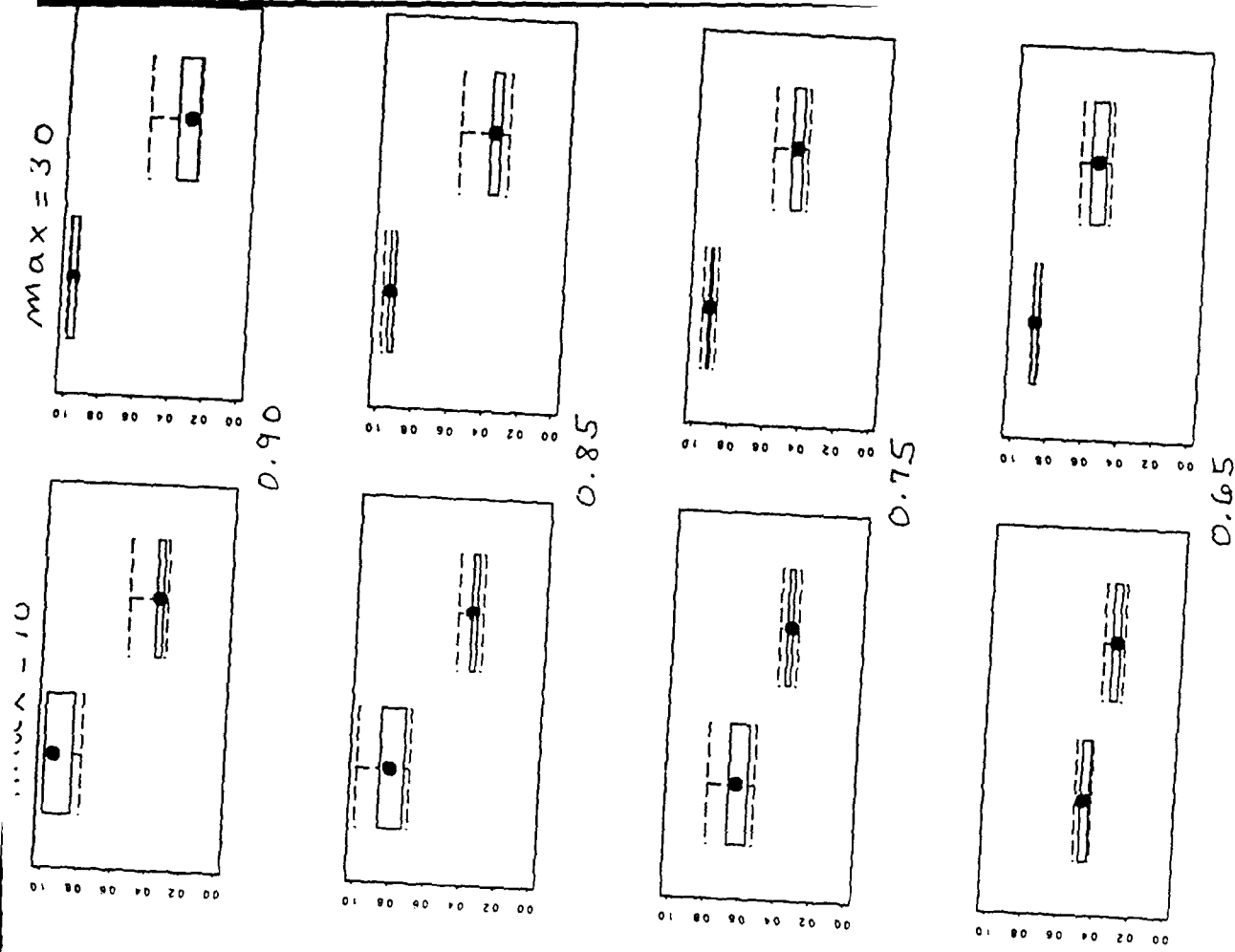
$$\sigma = 0.08$$

$$\text{ave } |\text{corr} \{f_m(x)\}_I^M| =$$

0.90 0.85 0.75 0.65

max. # of basis fun's (MARS) =

10 30



Mixtures of Experts Revisited

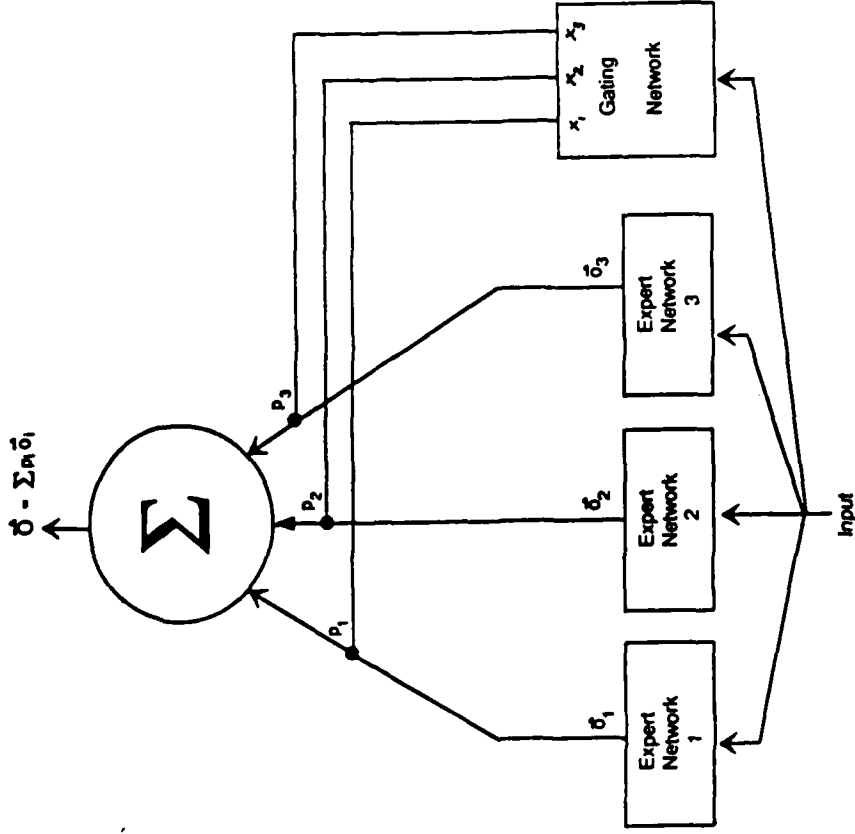
Steven J. Nowlan
Synaptics

Mixture of Experts

A modular architecture which decomposes a problem into a set of quasi-independent tasks:

- Each module is a local expert which learns to perform one or more tasks. Experts compete for the right to learn each training pattern.
- Gating network mediates the competition among experts. Like a manager it assigns different problems to different experts.
- Output is a **weighted average** of experts:

$$\vec{O} = \sum_i p_i \vec{o}_i$$



1. Objective: $\text{Max log } L = \sum_c \log P(\vec{d}_c | \vec{I}_c)$
2. Error for expert i : $E_i = p(E_i | \vec{d}_c)(\vec{d}_c - \vec{o}_i)$
3. Errors for gating net: $E_g(x_i) = p(E_i | \vec{d}_c) - p_i$

Mixture Equations

Objective:

$$\text{MAX log } L = \sum_c \log P(\vec{d}_c)$$

where

$$P(\vec{d}_c) = \sum_{i=1}^N p_i p(\vec{d}_c | i)$$

The probability of expert j generating the desired output is:

$$p(\vec{d}_c | j) = \frac{1}{K \sigma_j} e^{-\frac{\|\vec{d}_c - \vec{o}_j(\vec{I}_c)\|^2}{2\sigma_j^2}}$$

The gating network outputs are normalized with a "soft max":

$$p_j(\vec{I}_c) = \frac{e^{x_j^p}}{\sum_i e^{x_i^p}}$$

Expert network cost derivatives:

$$\frac{\partial \log L}{\partial \vec{o}_j} = \frac{1}{\sigma_j^2} \sum_c p(j | \vec{d}_c) (\vec{d}_c - \vec{o}_j)$$

where

$$p(j | \vec{d}_c) = \frac{p_j p(\vec{d}_c | j)}{\sum_i p_i p(\vec{d}_c | i)}$$

Gating network cost derivatives:

$$\frac{\partial \log L}{\partial x_j^p} = \sum_c p(j | \vec{d}_c) - p_j$$

Approaches to Combining Networks

	Static	Dynamic
Averaging	Committees	Mixture of Experts
	GEM	Meta-Pi
Stacking	Clustering Networks	
	Stacked Generalizers Boosting	Constructive M.E.?

Mixture of Experts:

- Output = a convex combination of expert outputs.
- Weights $\rightarrow$ computed dynamically as a function of input to gating net.
- Gating net with only thresholds:

$$w_j = \langle p(E_j | d) \rangle$$

Maximizing Effectiveness

Like other averaging schemes, the mixture of experts can be more effective by attempting to minimize the correlation between expert errors.

Examples:

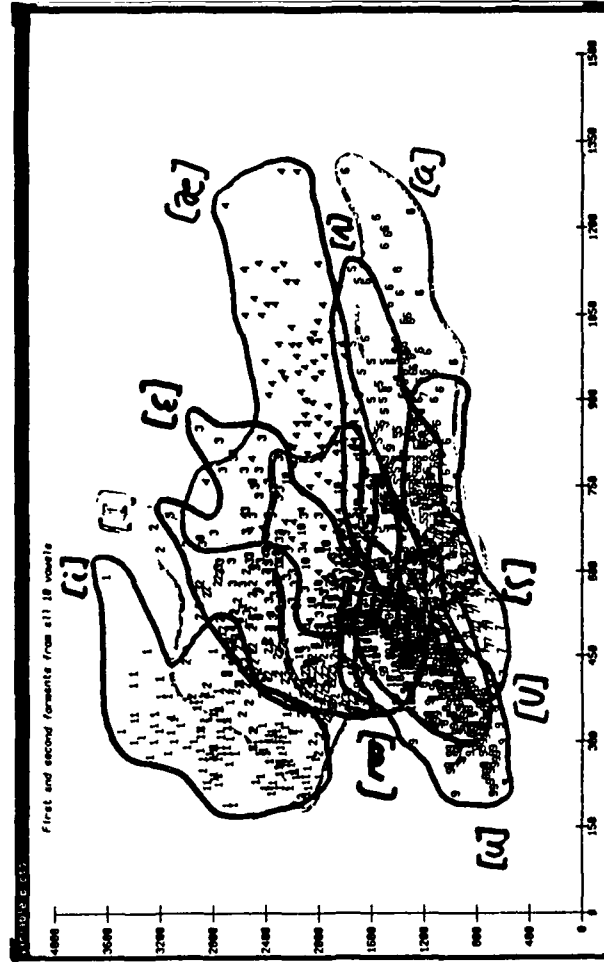
1. Gating network input is different from expert network input.
2. Expert network inputs are different (fusion).

Experiments on 10 Vowel Data Set

Data – First and second formants from 2 repetitions of 10 different vowels from 75 speakers (32 Male, 28 Female, 15 Children). From Peterson & Barney, 1952.

Systems Simulated

- A variety of multi-layer back propagation (BP) networks for benchmarking.
- Mixtures of experts where each expert received the formants as input, but the input to the gating network varied:
 - formant values
 - speaker identity
 - male, female or child (MFC)
 - MFC + min and max values of F1 and F2 for each speaker
 - min and max values of F1 and F2 for each speaker



Word List for Vowel Study

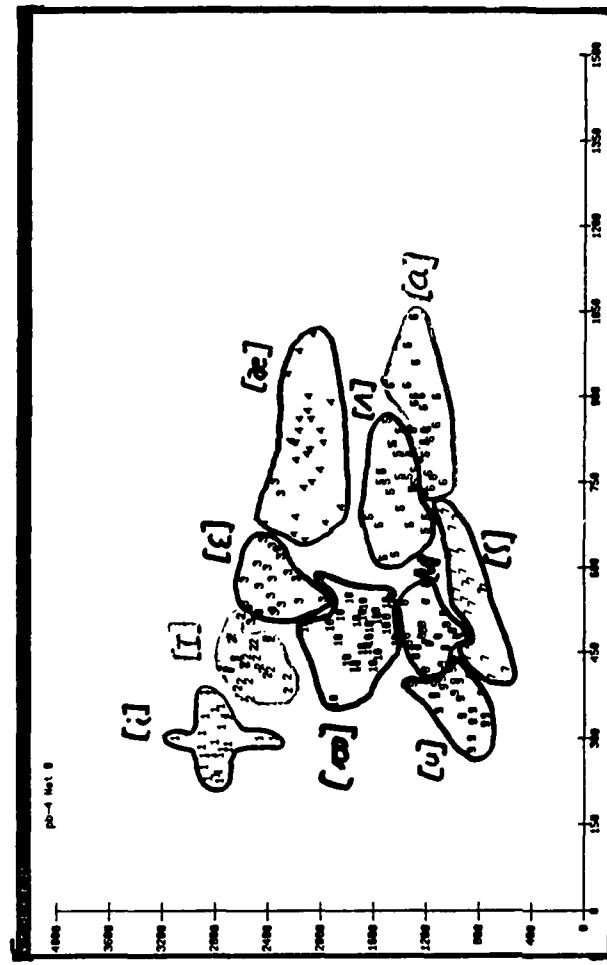
Word	Vowel Symbol	Class Number
feed	[i]	1
hid	[I]	2
head	[ε]	3
had	[æ]	4
hud	[Λ]	5
hod	[a]	6
hawed	[ɜ]	7
hood	[U]	8
who'd	[u]	9
heard	[ɰ]	10

Comparison to Single BP Network Generalization on Test Data

Type of Input	Mixture Error % (Test)	BP Error % (Test)	Sig.(p)
Formants only	15.1 ± 0.9	23.3 ± 1.2	≥ 0.9999
Form. + Speaker ID	6.4 ± 1.3	-	≥ 0.9999
Form. + MFC	13.5 ± 0.6	18.4 ± 1.1	≥ 0.9999
Form. + MFC + Range	6.2 ± 0.9	16.1 ± 1.0	≥ 0.9999
Form. + Range	12.8 ± 0.9	16.2 ± 0.8	> 0.9999

Example Speaker Decomposition

Expert #	% Male	% Female	% Child	% Total
0	0.0	0.0	6.7	1.3
4	3.1	3.6	0.0	2.7
5	84.4	17.8	0.0	42.7
7	9.4	7.1	6.7	8.0
8	3.1	42.9	0.0	17.3
9	0.0	28.6	86.7	28.0

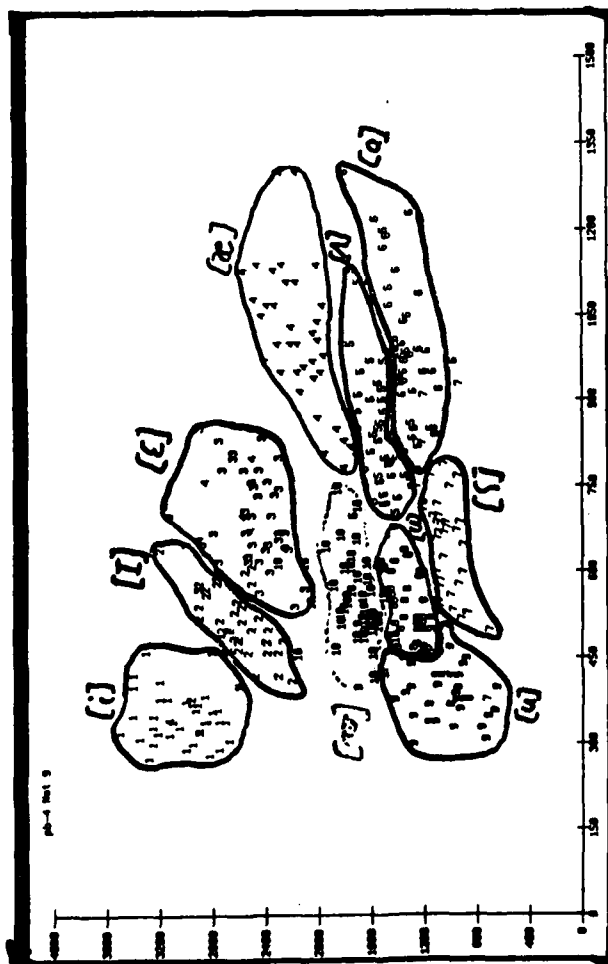


Hand Tracking

Problem: Tracking a hand through a sequence of video frames.

Two sources of input:

1. Intensity images
2. Difference images



Network Architecture

Two experts, each 3 layer convolutional networks:

1. Receives only intensity image as input.
2. Receives only difference image as input.

Gating network:

- 3 layer network, convolutional hidden, non-convolutional output
- low resolution versions of intensity and difference images as input
- simple global features can effectively weight two experts

Extensions

1. Hierarchical Decomposition – Trees of Experts (Jordan & Jacobs 92)

$$P(d|x) = \sum_i g_i \sum_j g_{ji} P(d|x, E_{ij})$$

- EM based IWRLS algorithm for linear experts
2. Support regions for multiple hypotheses
 - one set of experts
 - multiple gating networks
 - multiple weighted averages of expert outputs

$$L = \prod_h P(d_h|x)$$

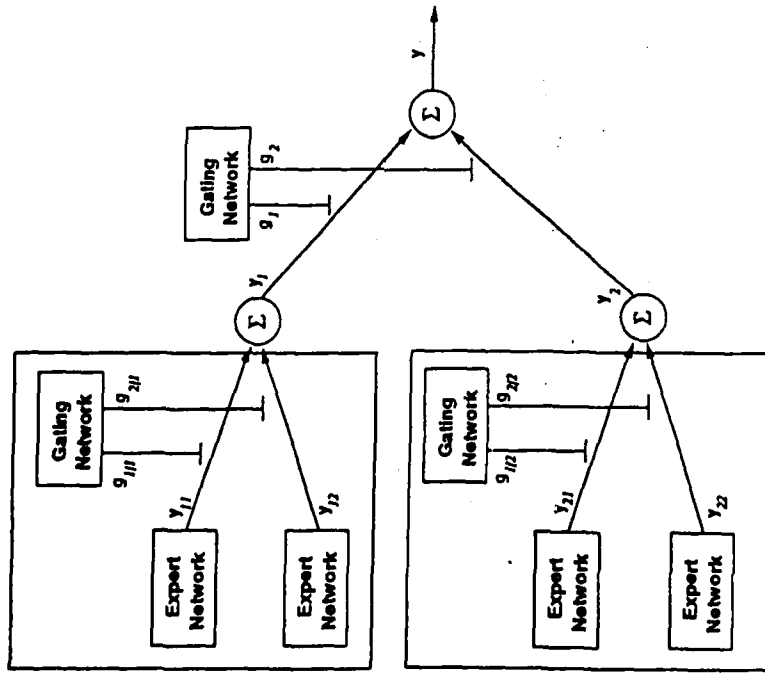
$$P(d_h|x) = \sum_i g_{hi} P(d_h|x, E_i)$$

3. Gating based on state trajectories
 - HMM like extension to M.E.

$$L(y_1 \dots y_t | x_1 \dots x_t) =$$

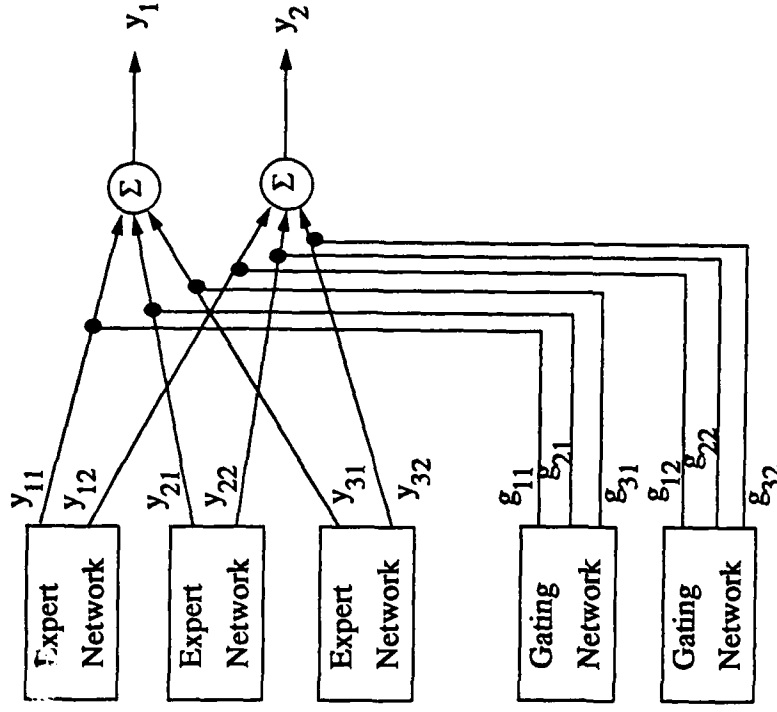
$$\prod_t \sum_k P(y_t | s_t = k, x_t) P(s_t = k | y_1 \dots y_{t-1}, x_1 \dots x_t)$$

Hierarchy of Experts



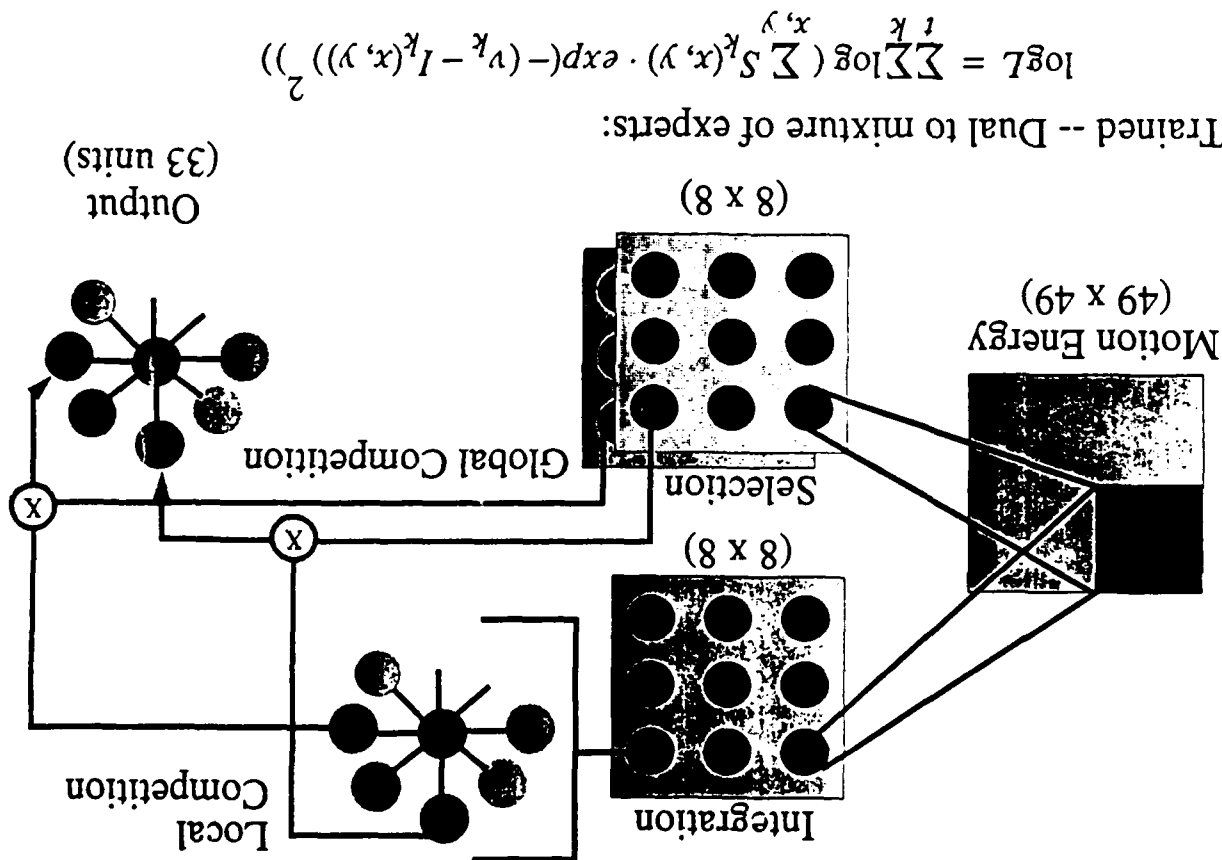
1. Error for expert ij : $E_i = p(b_i|\vec{d}_c)p(b_{ij}|\vec{d}_c)(\vec{d}_c - \vec{o}_i)$
2. Errors for gating net ij : $E_g(x_{ij}) = p(b_{ij}|\vec{d}_c) - g_{ij}$
3. Errors for gating net i : $E_g(x_i) = p(b_i|\vec{d}_c) - g$

Multiple Support Sets



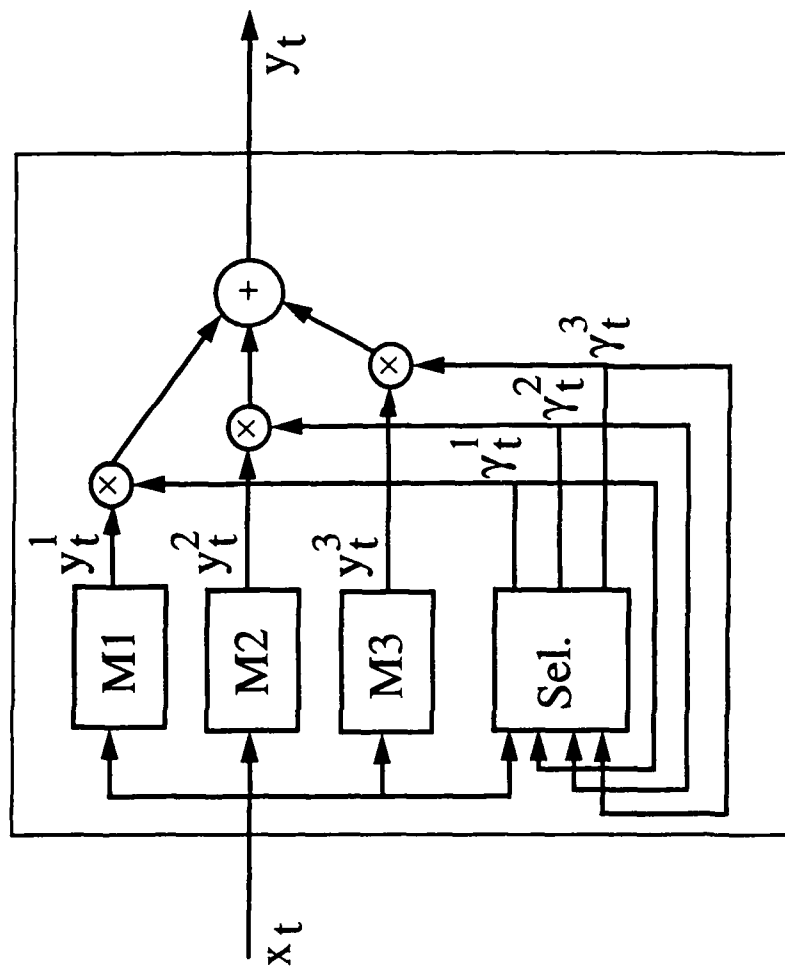
1. Error for expert y_{ij} : $E_{ij} = p(e_i|d_j)(d_j - y_{ij})$
2. Errors for gating net g_{ij} : $E_g(x_{ij}) = p(e_i|d_j) - g_{ij}$

INTEGRATION AND SELECTION



Trained -- Dual to mixture of experts:

$$\log L = \sum_{i=1}^I \log \left(\sum_{k=1}^K S^k(x, y) \cdot \exp(-(v_k - I^k(x, y))^2) \right)$$



$$\begin{aligned} L(y_1, \dots, y_t | x_1, \dots, x_t) &= \prod_i \sum_k P(y_i | s_i = k, x_i) P(s_i = k | y_1, \dots, y_{i-1}, x_1, \dots, x_t) \\ &= \prod_i \sum_k b_i^k \gamma_i^k \end{aligned}$$

$$P(y_t | y_1, \dots, y_{t-1}, x_1, \dots, x_t) = \sum_k b_t^k \gamma_t^k$$

$$b_t^k = f_{w_k}(x_t)$$

$$\gamma_t^k = f_{w_{\gamma}}(x_t, \{\gamma_{t-1}^k\})$$

Models / Controllers

Open Problems

$$b_t^k = \frac{1}{\sqrt{2\pi}\sigma} e^{-(w_t - \hat{w}_t)^2 / 2\sigma^2} \quad L_t = \sum_k b_t^k \gamma_t^k$$

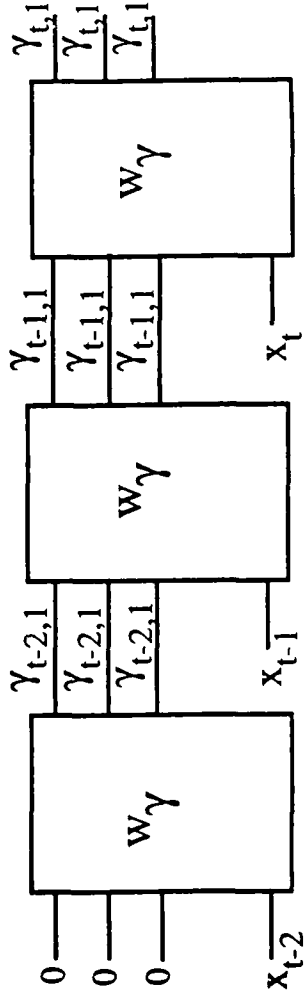
$$\frac{\partial \log L}{\partial w_k} = K \sum_i \underbrace{b_i^k \gamma_i^k}_{\beta_i^k} (y_i - \hat{y}_i^k) \frac{\partial y_i^k}{\partial w_k}$$

Selector

$$\gamma_t^k = \frac{\exp g_t^k}{\sum_j \exp g_t^j}$$

$$\frac{\partial \log L}{\partial w_\gamma} = \sum_i (\beta_i^k - \gamma_i^k) \left(\sum_{r=0}^T \sum_j \frac{\partial g_{i-r}^k}{\partial g_{i-r}^j} \frac{\partial y_{i-r}^k}{\partial w_\gamma} \right)$$

1. Can theoretical results on averaging be extended to schemes like the M.E.?
2. Are there dynamic equivalents of stacking and boosting?
3. Hybrid architectures
i.e. Basis Fn gating + MLP experts



**BOOSTING
AND OTHER ENSEMBLE METHODS**

**HARRIS DRUCKER
CORINNA CORTES
L.D. JACKEL
YANN LECUN
VLADIMIR VAPNIK**

AT&T BELL LABS

QUESTIONS ASKED:

**FOR A GIVEN TRAINING SET SIZE:
WHICH ALGORITHM IS BEST ?**

**FOR A GIVEN COMPUTATIONAL COST:
WHICH ALGORITHM IS BEST?**

figure-1a talk

figure-2 2a talk

WE WILL SHOW THAT

BOOSTING IS THE BEST ALGORITHM

**GENERALIZATION IS BETTER IF WE TRAIN
ON A SUBSET OF THE TRAINING POOL
AND DISCARD THE REST**

ADDITION IS BETTER THAN VOTING

TRAINING SET

EXAMINED BUT DISCARDED	USED FOR TRAINING = COMPUTATIONAL COST
-----------------------------------	---

figure-2.1a talk

figure-2.3a

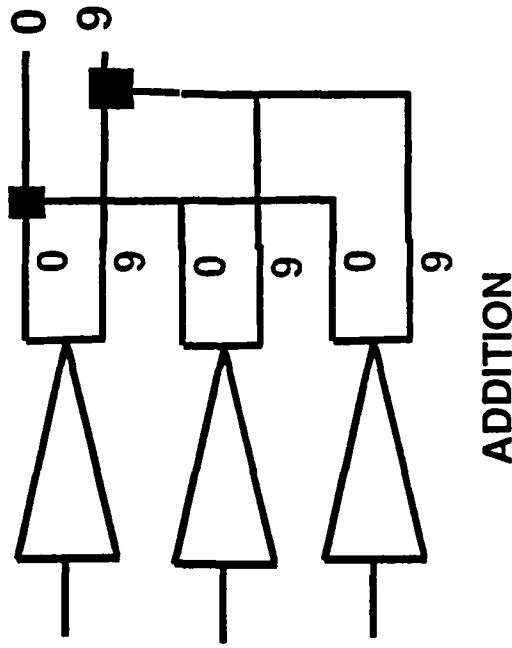
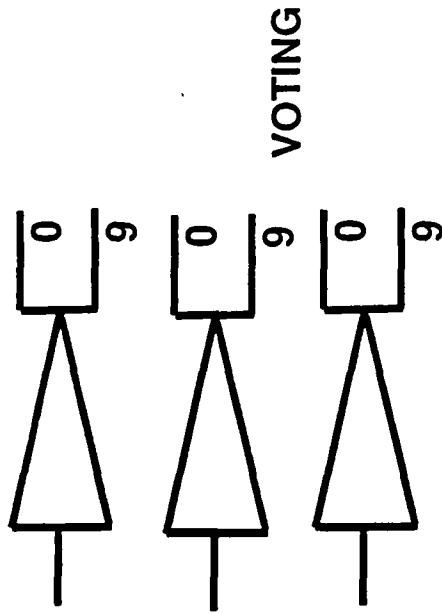


figure-2.4a.talk

DATABASE:

120,000 DIGITS FROM NIST
2000 FOR TESTING
10X10 INPUT SPACE

FOR each algorithm (4 of them)

FOR each architecture (3 of them)

FOR each training set size

REPEAT 10 times

pick training set

randomize weights

train to mse minimum

END_repeat

get average test and train error

END_FOR

END_FOR

END FOR

figure-3a.talk

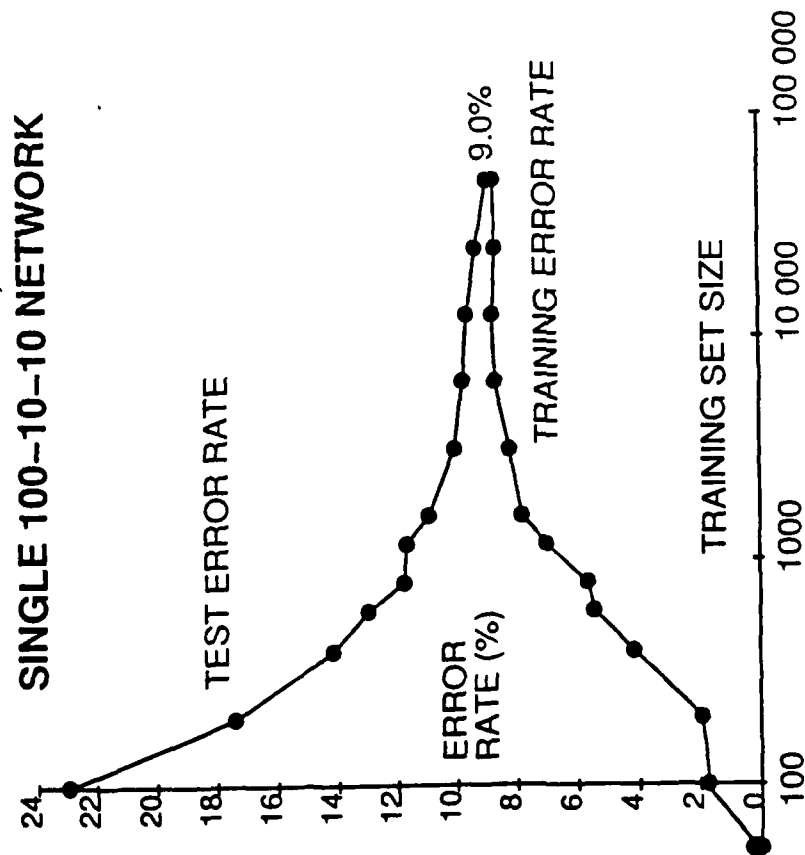


Figure-4a talk

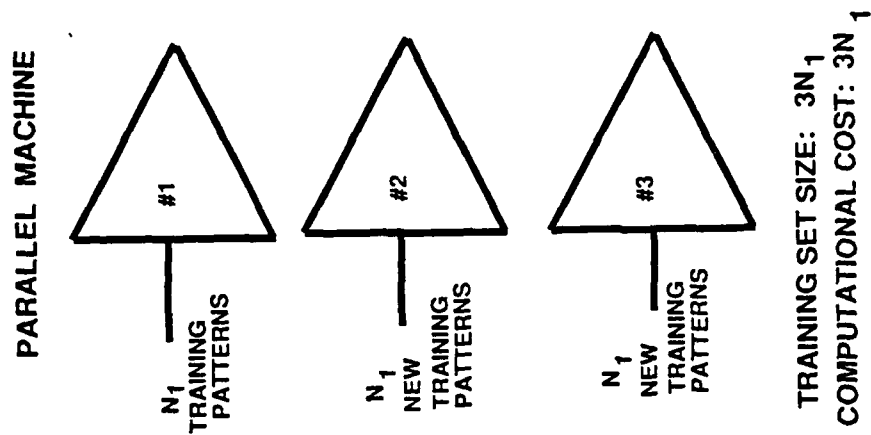


Fig-5 talk

ORIGINAL BOOSTING

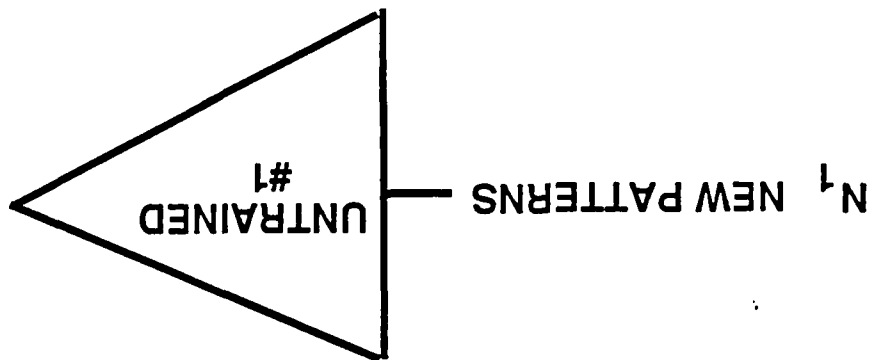


fig-6 talk

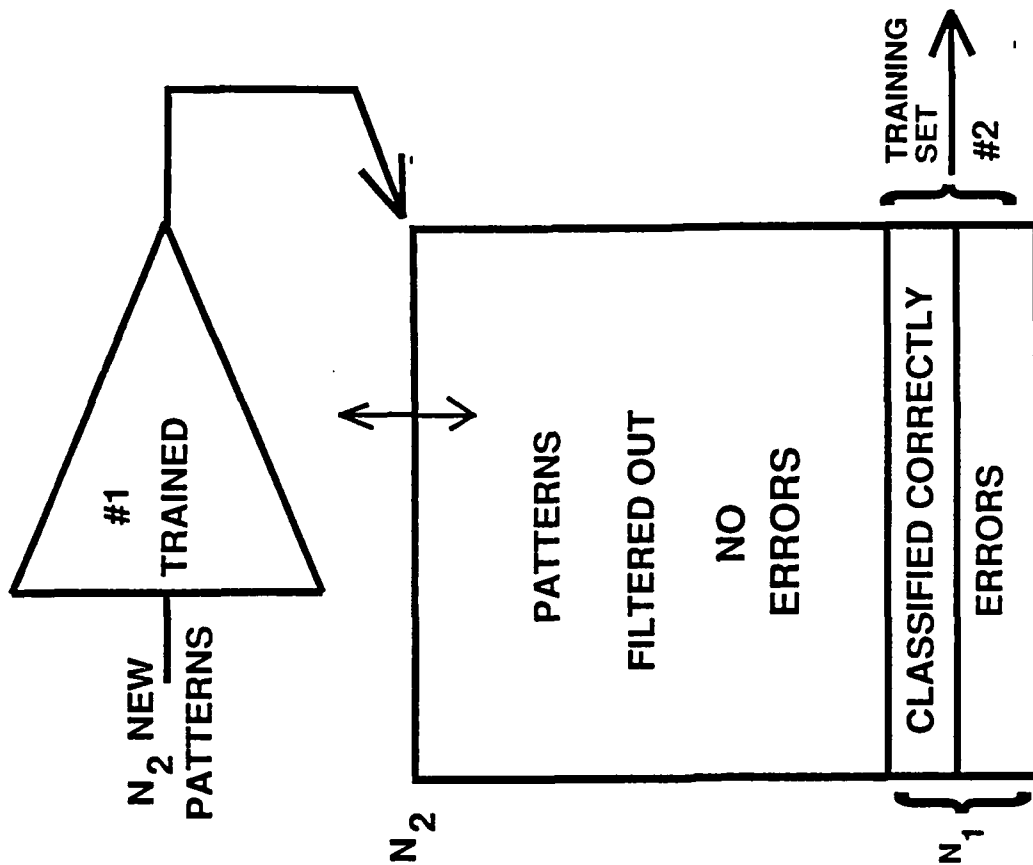


figure-7a talk

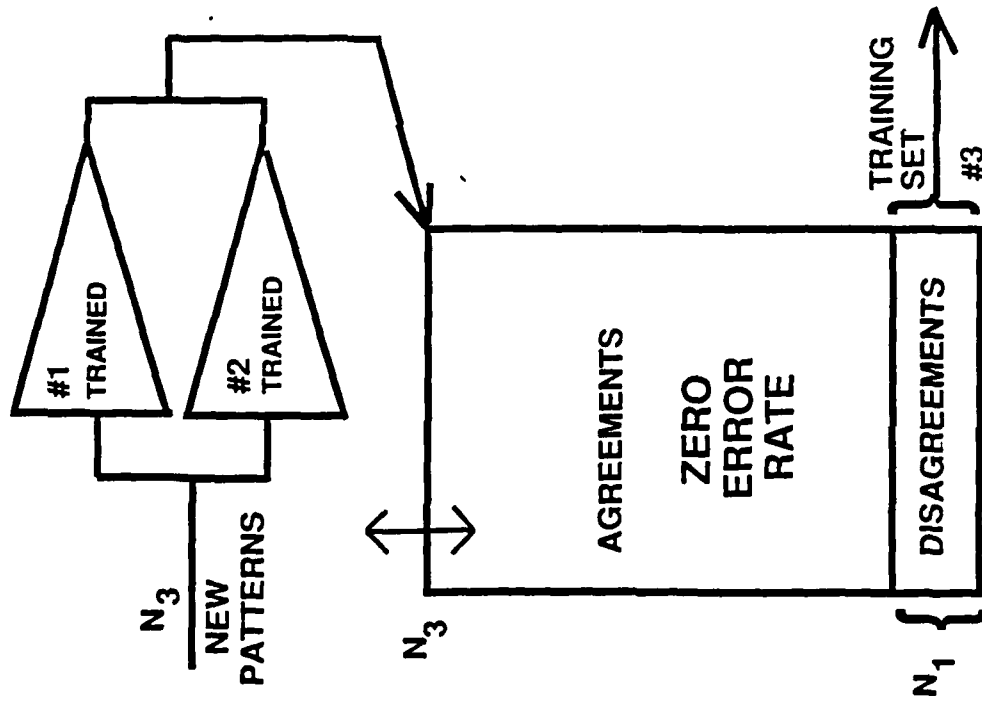
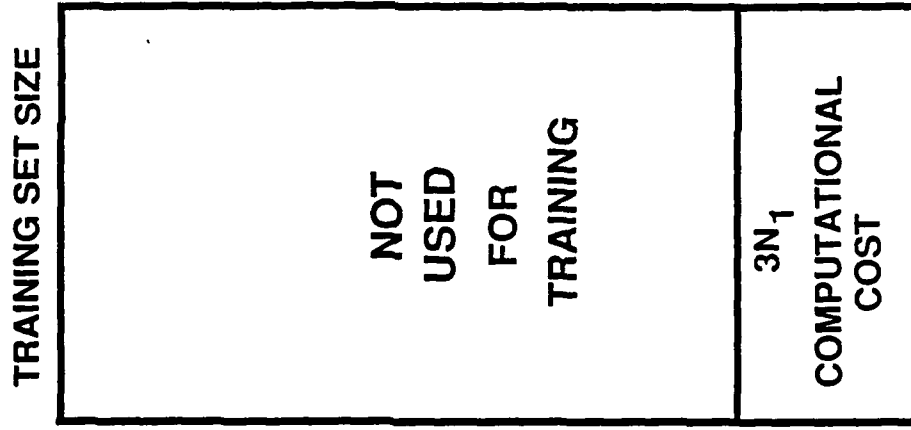


figure-8a talk



ORIGINAL BOOSTING

figure-9a talk

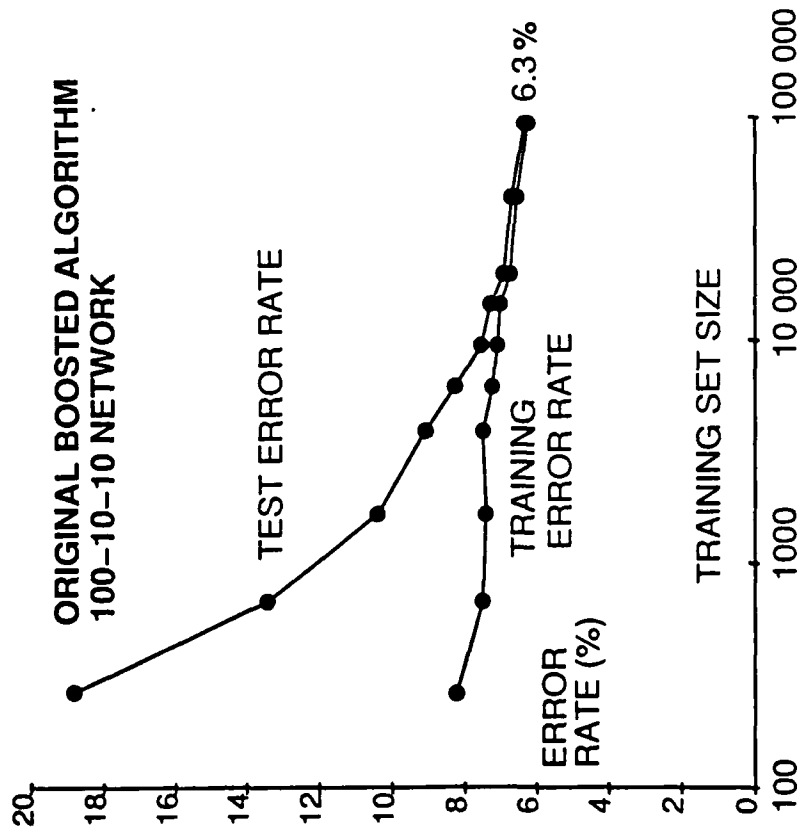


figure-10.tsk

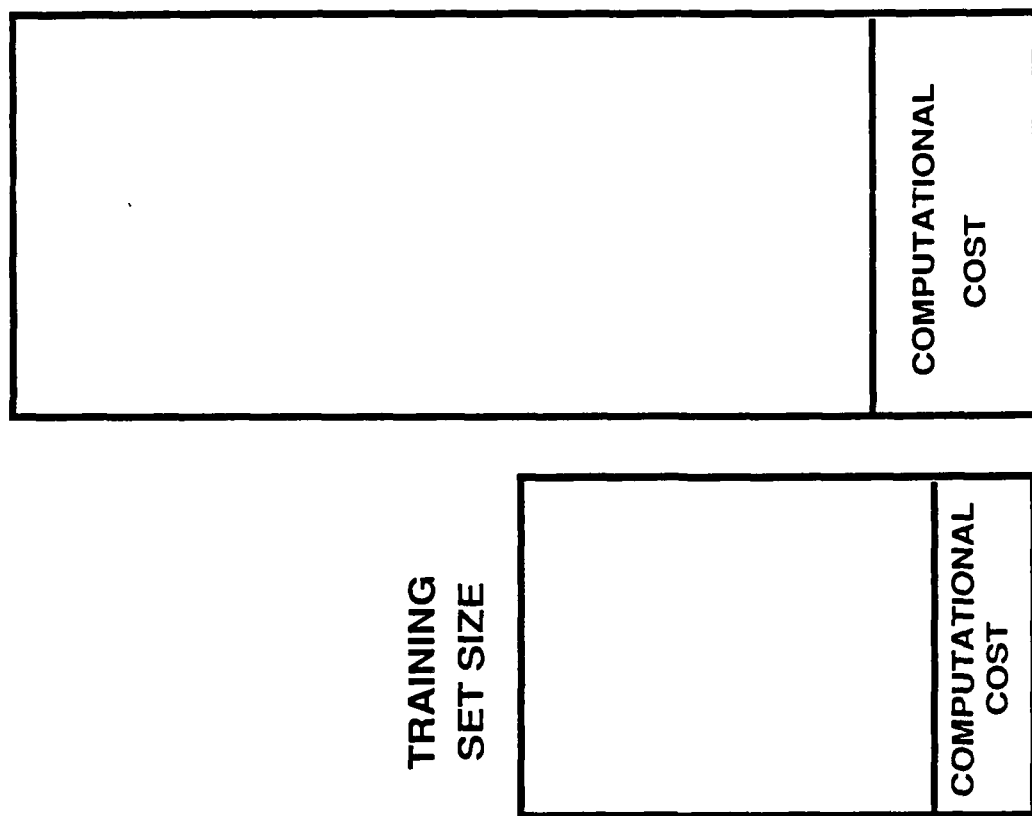


figure-11.tsk

MODIFIED BOOSTING

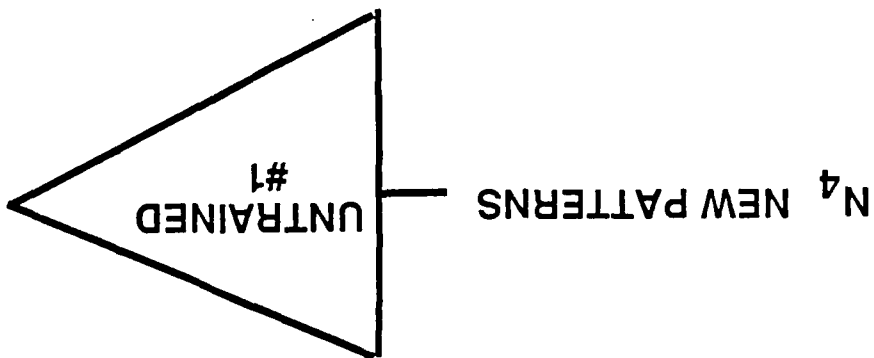


Figure-12 talk

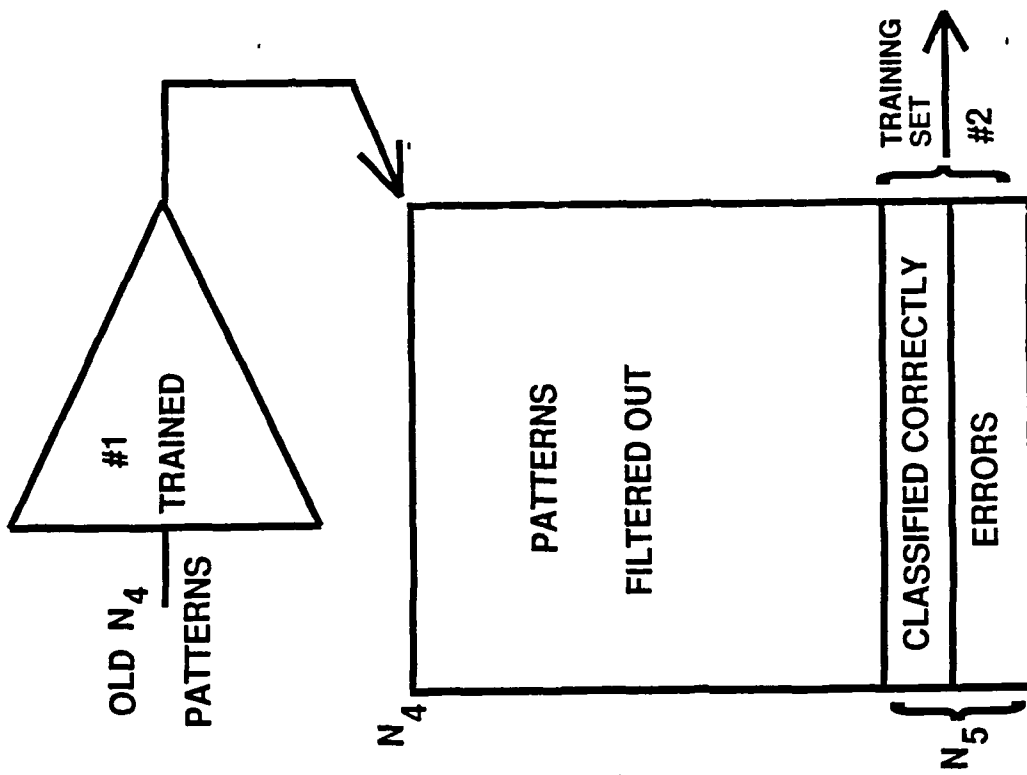


Figure-13 talk

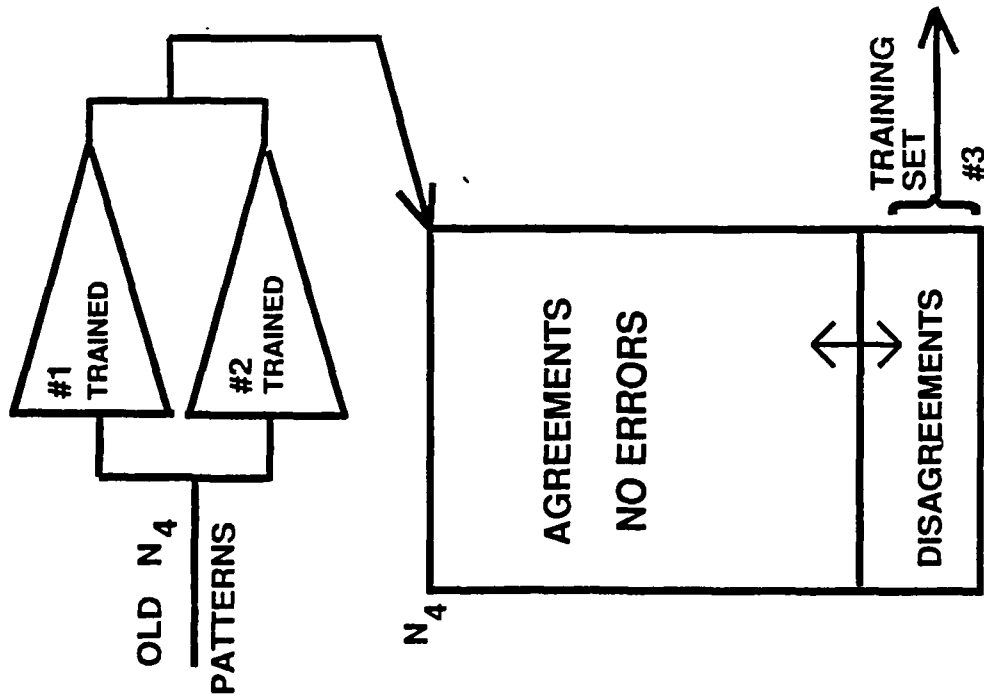


figure-14a talk

COMPUTATIONAL COST

SUBSET OF N_4 USED TO TRAIN #3
SUBSET OF N_4 USED TO TRAIN #2
TRAINING SET SIZE (N_4) USED TO TRAIN #1

MODIFIED BOOSTING

figure-15a talk

100-10-10 NETWORK

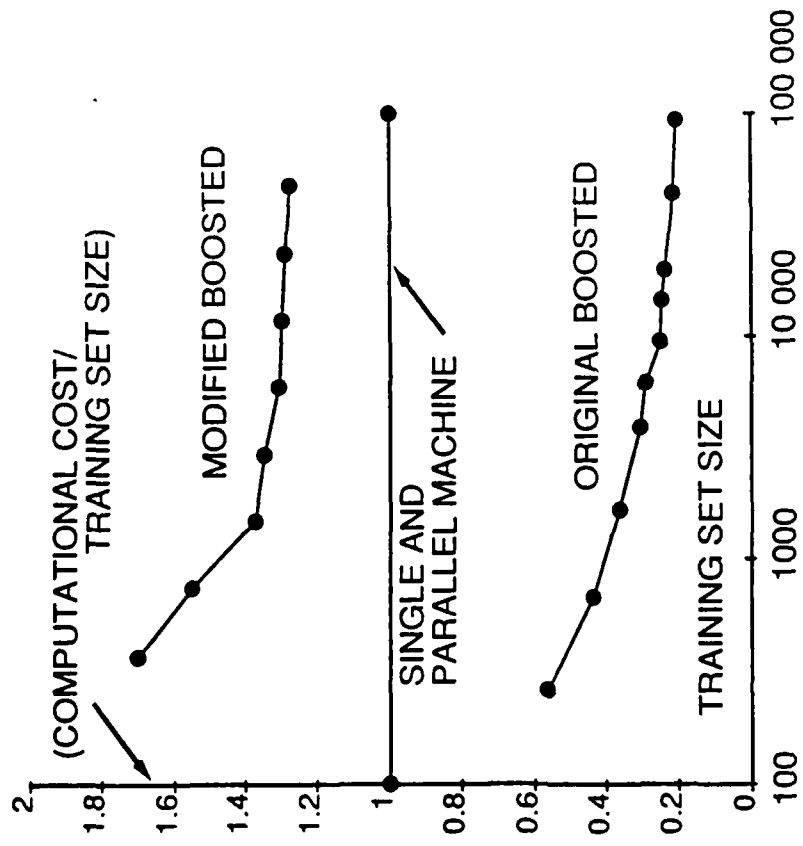


figure-16 talk

100-10-10 NETWORK

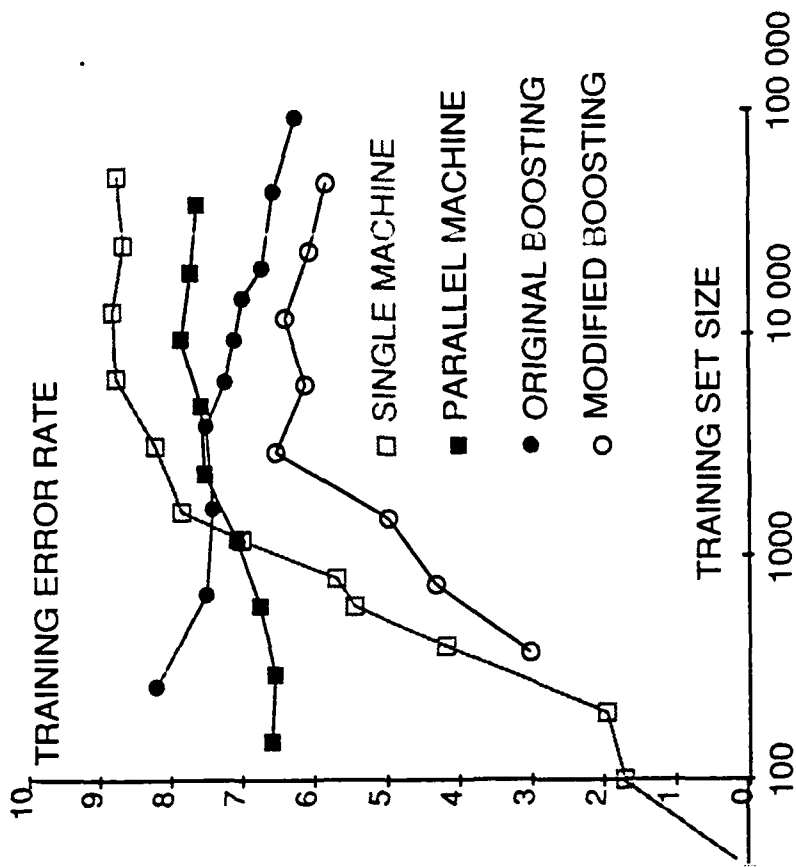


figure-17 talk
figure-17a talk

100-10-10 NETWORK

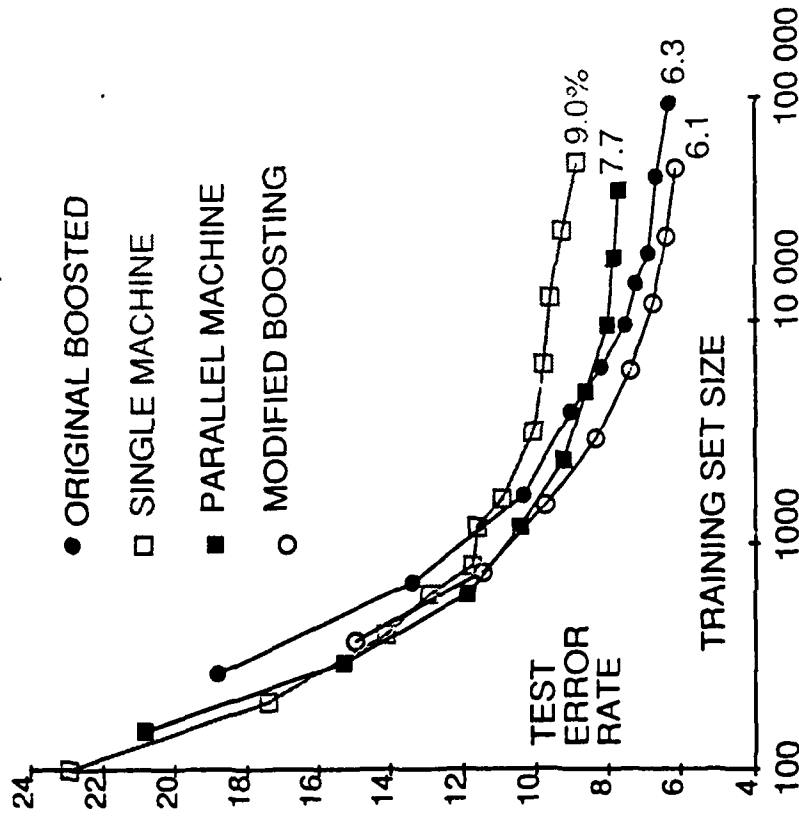


figure - 18a talk

100-10-10 NETWORK

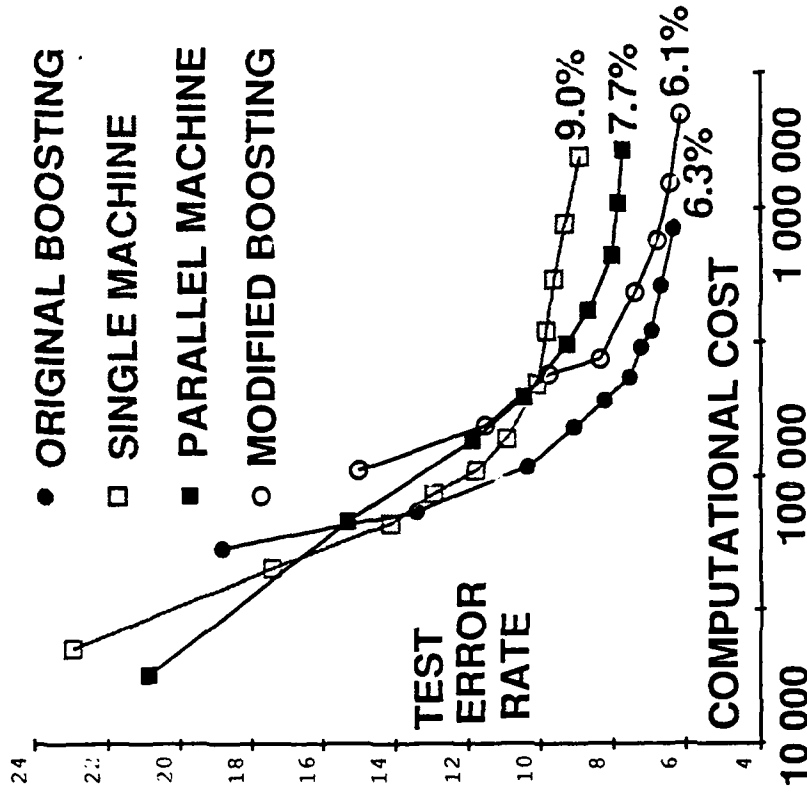


figure - 20a talk

100-10-10 NETWORK

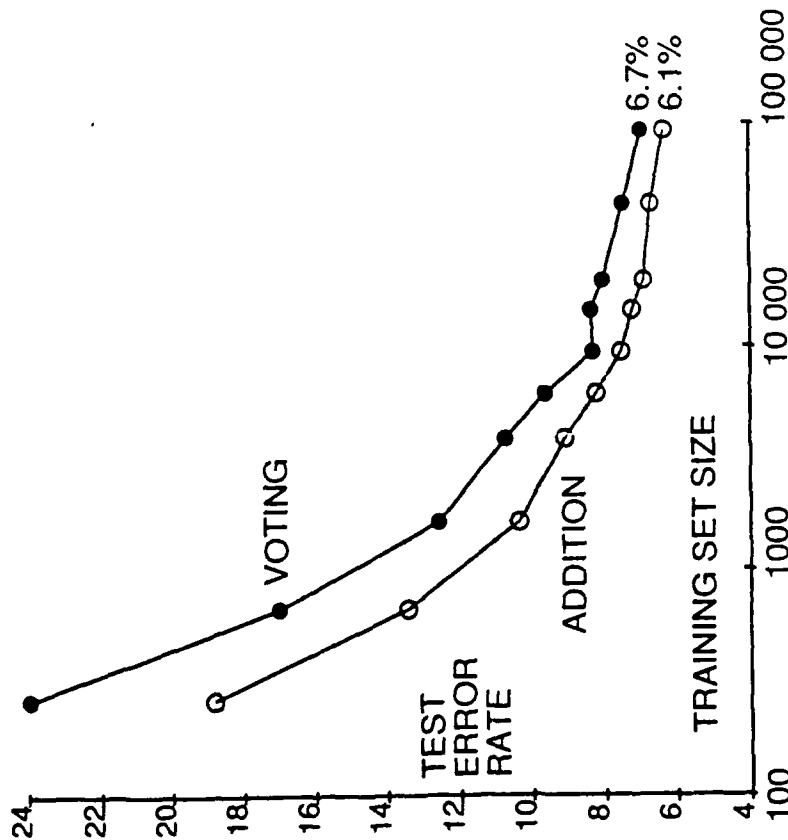


figure-21 talk
figure-21a talk

A LARGER NETWORK
13,561 NEURONS
17,132 WEIGHTS
260,180 CONNECTIONS

TRAINING SET SIZE: 2.5 MILLION
COMPUTATIONAL COST: 180,000

ONE NETWORK: 113 ERRORS/10,000

BOOSTED ENSEMBLE: 70 ERRORS/10,000

figure-27a talk

WE SHOWED THAT

BOOSTING IS THE BEST ALGORITHM

GENERALIZATION IS BETTER IF PART OF
THE TRAINING POOL IS DISCARDED

ADDITION IS BETTER THAN VOTING

FOR THE ORGINAL BOOSTED ENSEMBLE
THE TRAINING ERROR RATE DECREASES
WITH INCREASING TRAINING SET SIZE

REFERENCES:

SCHAPIRE: THE STRENGTH OF WEAK
LEARNABILITY, *MACHINE LEARNING*,
VOL 2, NO. 2, 197-227

DRUCKER, SCHAPIRE, AND SIMARD:
IMPROVING PERFORMANCE IN
NEURAL NETWORKS USING A
BOOSTING ALGORITHM,
NIPS 5, 42-49

A Commercial Application: Extracting Document Content from Images

Christopher L. Scofield
Harry Chang
Ed Collins



Structure of the Problem

- OCR is not really a problem of character recognition. It is really language processing from images:

Character context drives segmentation:



Lexical context drives character interpretation:



Structure of the Problem

Lexical context drives character interpretation:

clone < clone?
done?

Application specific rules drive interpretation:

54Y293?
54Y2A3?
J4Y2A3?



Nestor, Inc. _____

Structure of the Problem

Document structure drives syntactic and lexical possibilities:

VEGA Resources
2931 Loveland
TINSEL WA 02931

Company Name

Street number Street name

City State Zip



Nestor, Inc. _____

What are the possible approaches?

- **Single Network Architecture**

- » [Keeler90]: Combined segmentation and recognition;
- » [Fontaine92]: RNN trained on pixel-columns

- **Pluses:**

- » Makes no assumption about structure of problem
- » Automatically trains each part of problem

- **Minuses:**

- » Scaling problem
- » Lack of modularity: application dependent



Possible methods

- **Multiple Network Architecture**

- » [Gouin92]: Neural network segmentation for map processing;
- » [Scofield92]: Context-driven segmentation, recognition

- **Pluses:**

- » Each module can be built in a minimal fashion
- » Only some parts need to be changed for new applications

- **Minuses:**

- » Assumes prior knowledge
- » Must be assembled in a piecewise fashion
- » Credit assignment problem



- **Task decomposition:**



- **Neural network is used to assemble a tree representation of the image:**

-

Segmentation: Step 1

- 3-layer BPN trained to classify blobs into:

"Character"

"Noise"

"Mixed"



- Use connectivity analysis features [Hu62] including:

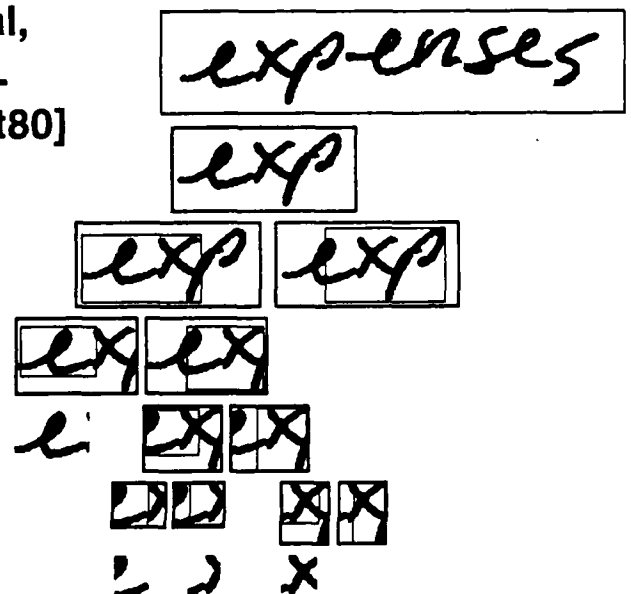
area, perimeter, number of holes,
area of holes, principle moments,
aspect ratio, etc.



Nestor, Inc.

Segmentation: Step 2

- "C", "N" are terminal,
"M" parsed with quad-
tree analysis [Samet80]
Re-classified at each
step:



Nestor, Inc.

Segmentation: Step 2 (contd):

- Hierarchical Agglomerative Clustering [Duda72] groups terminal nodes; re-classified to ensure still terminal:



Step 3:

- List of segmentation alternatives compiled for classification into characters
 - Multiple "cuts" of characters provided for later analysis:



Nestor, Inc. _____

Segmentation Accuracy

- Test set consists of 5,654 HP/MP characters in 1,236 words (46% HP) selected from 53 real-world documents
- HP data consists of live forms with constrained HP, unconstrained HP, run-on HP and some cursive
- Character segmentation accuracy:

(First choice correctly segmented)

Segmentation Network

"Blob" features (7-10-3)

Correct Incorrect

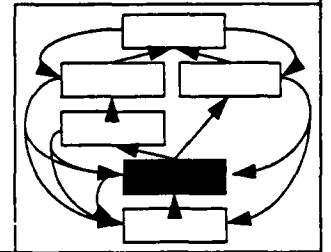
92.7 7.3



Nestor, Inc. _____

Character Recognition: Overview

- Segmentation alternatives are processed for character class
- Use two static feature sets: (cf - [LeCun90])
- Three-layer, feedforward BPNs are used as estimators of *a posteriori* probabilities
- We have employed three types of hybrid networks:
 - Glue networks [Waibel88]
 - Parallel Experts [Reilly87]
 - Hierarchical Filters [Reilly87]



Nestor, Inc.

Character Recognition: Feature Extraction

- Segmentation alternatives are converted to grey-level: gaussian kernel estimated from line widths
- Pixel Feature Set:
 - Pixel map is sub-sampled with grid producing coarse map (100-element grid)
- Edge Feature Set:
 - Edge-map produced from grey-level gradient estimation [Roberts65] (4 edge directions)
 - Edge map is sub-sampled with grid producing coarse edge map (30-element grid)



Nestor, Inc.

Character Recognition Classifiers

- Features used to train 3-layer, feedforward BPNs

<u>Data Set</u>	<u># Authors</u>	<u>Digits</u>	<u>Alpha U/LC</u>
Train: NIST 1,3; Propr.	2600	265,000	120,000
Test: Propr.		4,767	12,932

- In addition to using "Forced accuracy", can use a heuristic which models high cost of errors:

"Figure of Merit": $FM = 100 - 10(\%E) - \%R$

<u>Numeric Network</u>	<u>FM</u>	<u>Correct</u>	<u>Inc.</u>	<u>Reject</u>	<u>Forced</u>
Edge features (120-32-10)	95.15	97.04	0.21	2.75	99.01
Pixel features (100-45-10)	93.41	95.87	0.27	3.86	98.59

 *Nestor, Inc.*

Classifier Analysis

- Using "rule-of-thumb" $e = W/T$ [Baum89]:

Training set: $T = 265,000$
Edge Net: $W = 120 \times 32 + 32 \times 10 = 4,160$
Expected test error: $e = 1.7\%$

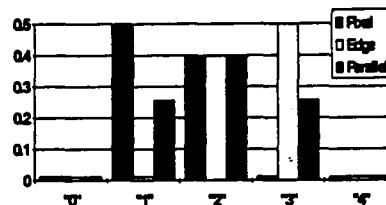
Pixel Net: $W = 100 \times 45 + 45 \times 10 = 4,950$
Expected test error: $e = 1.9\%$

 *Nestor, Inc.*

Character Recognition: Parallel Experts

- How to combine the results from two networks?
- Could vote if have many "experts". If only two, then average activation (probability) vectors:

$$P_i = 1/2(P_i^e + P_i^p)$$



Numeric Network

Edge features (120-32-10)

Pixel features (100-45-10)

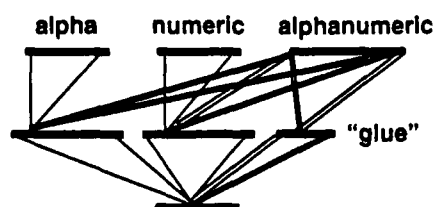
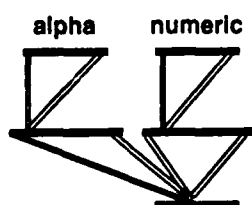
Parallel Nets:

FM	Correct	Inc.	Reject	Forced
95.15	97.04	0.21	2.75	99.01
93.41	95.87	0.27	3.86	98.59
97.25	98.20	0.10	1.70	99.39



Alphanumeric Character Recognition

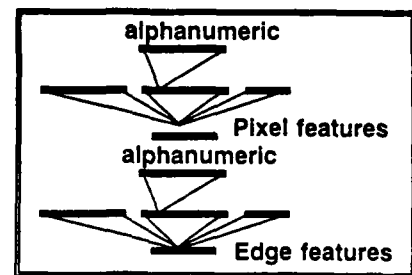
- Support full alphanumeric HP
- Natural decomposition into alpha and numeric subnets
- Use glue-net architecture [Waibel88]:
 - Trained 3-layer nets for alpha (u/l case) and numeric
 - Freeze middle-layer weights, route activations to output
 - Add-in new "glue" layer to resolve inter-class ambiguity
 - Train second layer of weights and all glue-cell weights



Alphanumeric Accuracy

• Results:

<u>Network Architecture</u>	<u>FM</u>	<u>Forced</u>	
		<u>Correct</u>	<u>Incorrect</u>
Numeric sub-net (120-32-10)	n/a	95.15	4.75
Alpha (U/L) sub-net (120-120-26)	n/a	91.41	8.59
Single Glue Net(1) (120-210-36)	51.24	89.24	10.76
Hierarchical (Super) Glue Net	54.04	90.80	9.20
Single Glue Net(2) (120-210-36)	54.46	89.98	10.02



 Nestor, Inc.

Glue Net Analysis

Digit set:	265,000
Digit Net:	$W = 120 \times 32 + 32 \times 10 = 4,160$
Expected test error:	$e = 1.7\%$
Alpha set:	120,000
Alpha sub-net:	$W = 120 \times 120 + 120 \times 26 = 18,720$
Expected test error:	$e = 15.6\%$
Full set:	385,000
Glue weights:	$W = 120 \times 58 + 210 \times 36 = 14,520$
Expected test error:	$e = 3.8\%$

 Nestor, Inc.

Character and Lexical Context

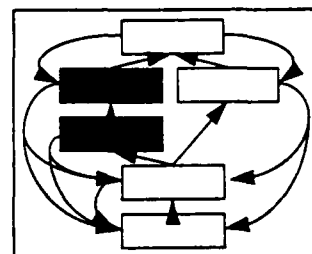
- Word recognition: determine the best string interpretation given all sources of knowledge:
 - segmentation alternatives
 - character recognition probabilities
 - character transition probabilities
 - lexical context

clone

el o n e
cl 0 m P
d u 0 0

DAVID

0 A V 1 D
D H U I P
O 4-2 L 0



Nestor, Inc.

Character and Lexical Context

- Use the Viterbi algorithm [Viterbi67, Forney73] to select the character string with maximum *a posteriori* probability
- Let maximization of word probability drive proper segmentation [Bozinovic82]
- Problem: VA can produce lexically incorrect strings. Post-processing with a dictionary can produce word which is not MAP.
- Solution: Use lexical context to trim paths from VA search ensuring that the final string is both MAP and lexically correct [Srihari83].



Nestor, Inc.

Application Context Processing

- Some applications are alphanumeric but not part of lexicons:

A1B00138

- Inter-character statistics are specialized, hard to learn without large set
- User-definable syntax selects which subnet to use for each character position, trims segmentation alternatives to match syntax

22152

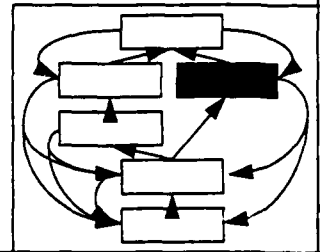
ZIP Code →

22152

22152



Nestor, Inc.



Status and the Future

- This architecture is the basis for the product NestorReader
- To be supported on Ni1000 neural net chip
- Extensible to character-based cursive recognition
 - Now developing much larger training and test sets for run-on HP and cursive
 - Developing stats. on character and segmentation accuracy due to character and lexical context



Nestor, Inc.

Forecasting the

U.S. Index of

Industrial Production

via

Neural Networks

John Mccody,

Uzi Levin, & Steve Rehrer

Oregon Graduate Institute

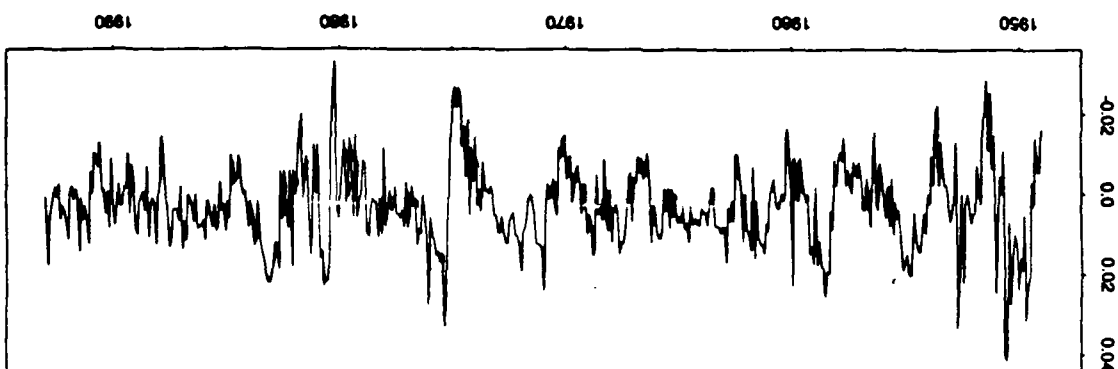
mcody@cse.ogi.edu

Outline

- I. Intro & Statement of Problem
Performance of Professional Economists
Difficulties:
 - Non-Normality
 - Non-Linearity
 - Non-Stationarity
- II. Performance of Linear Models
 - Univariate AR Models
 - Multivariate Regression & ARX Models
- III. Results for Multivariate NN Models
 - Regression Nets
 - Classification Nets
 - The "Noise/Nonstationarity" Trade-Off
 - Committees
- IV. Results for SPX

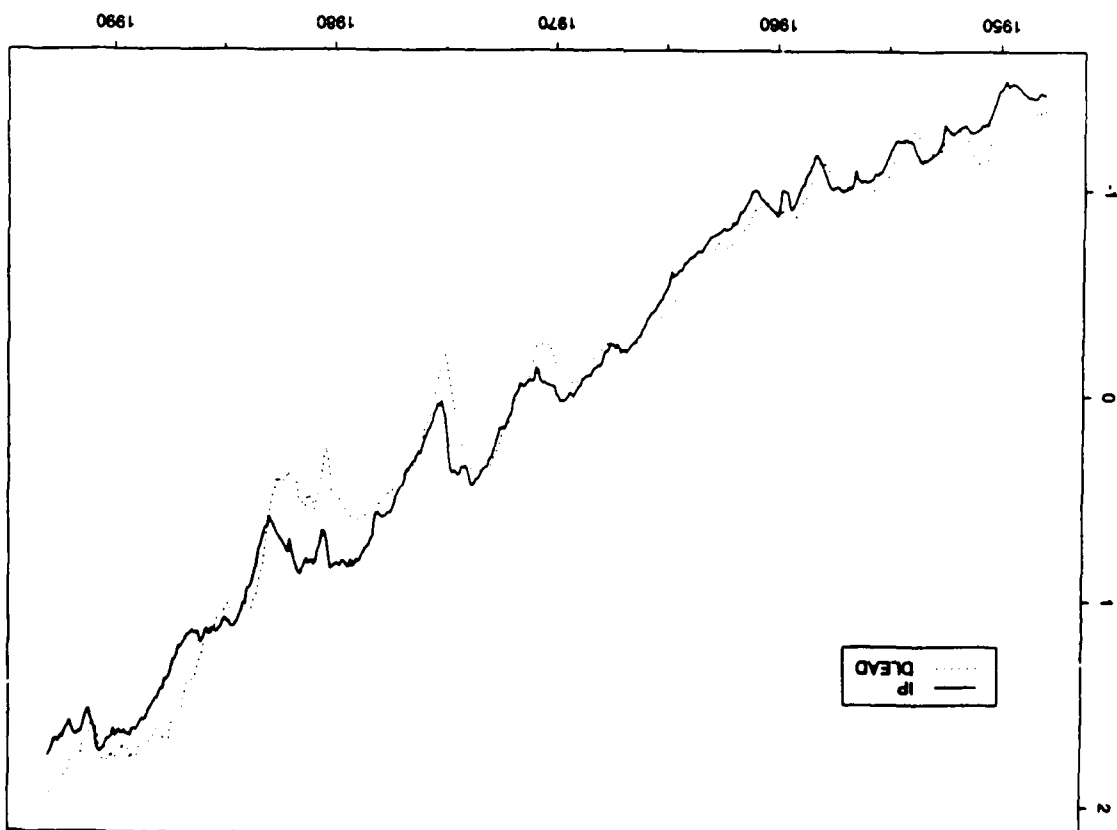


IP.PR1



DLEAD.PR1

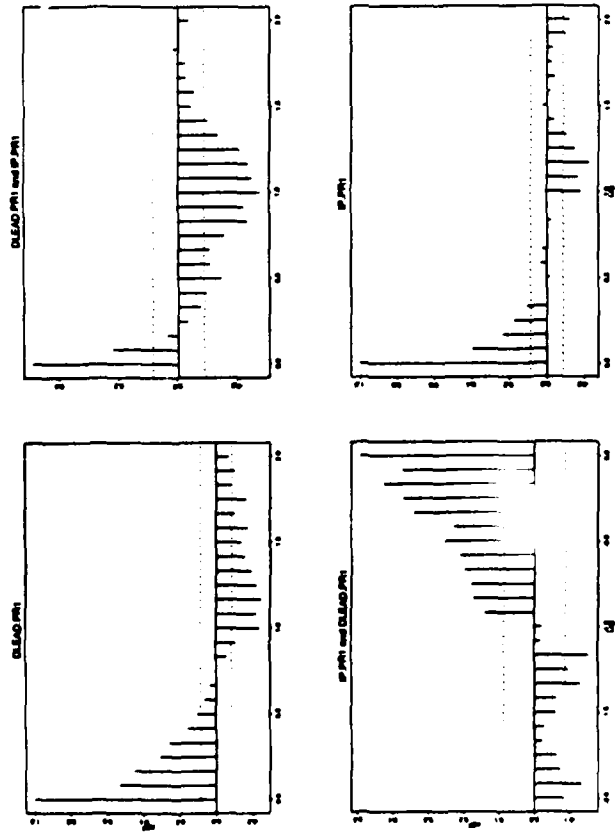
one month rates of returns



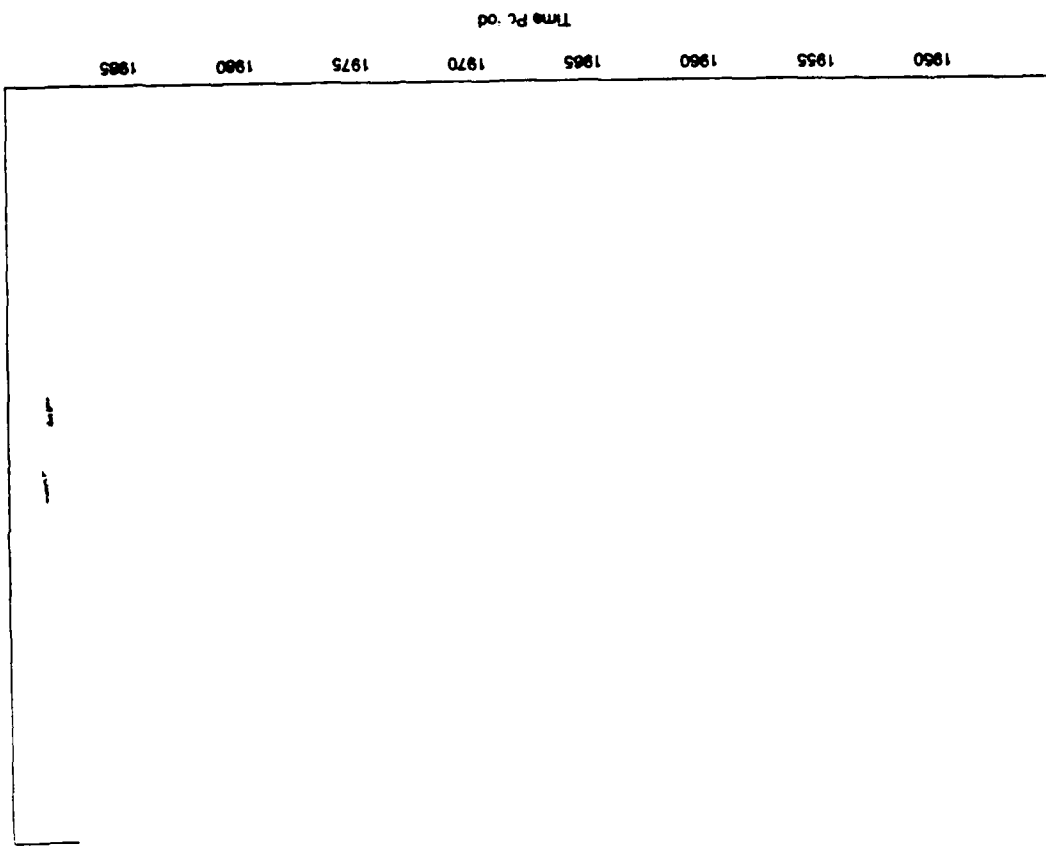
SCALED IP & DLEAD INDICES

The leading index leads industrial production.

Multivariate Series : DLEAD.IP.PR1



Note the nonstationarity of $\Delta \log(IP)$



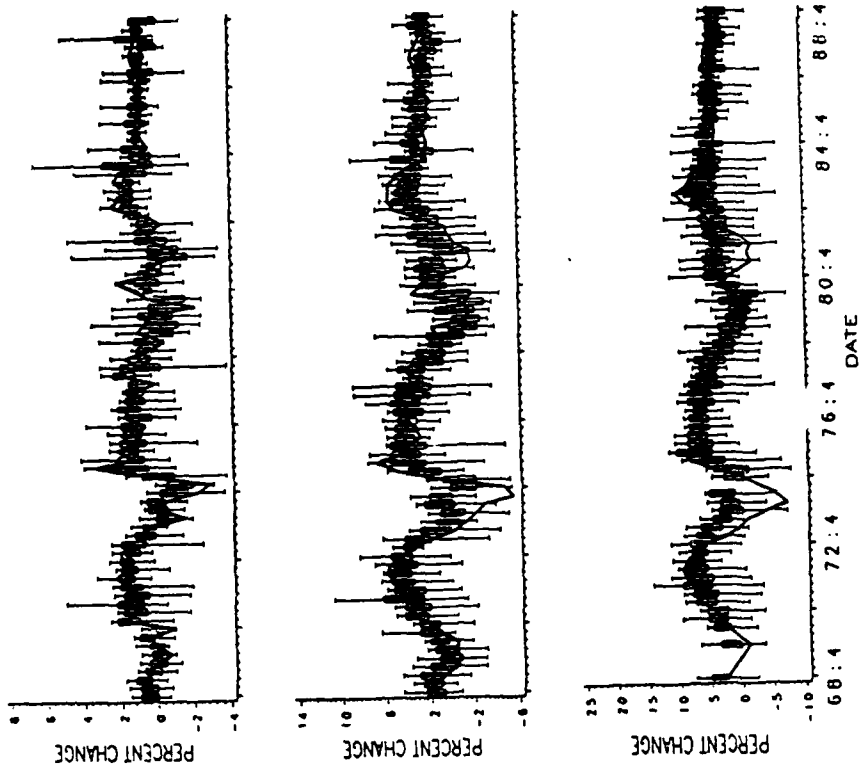
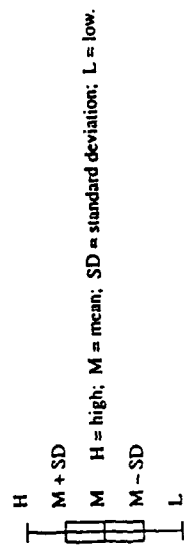
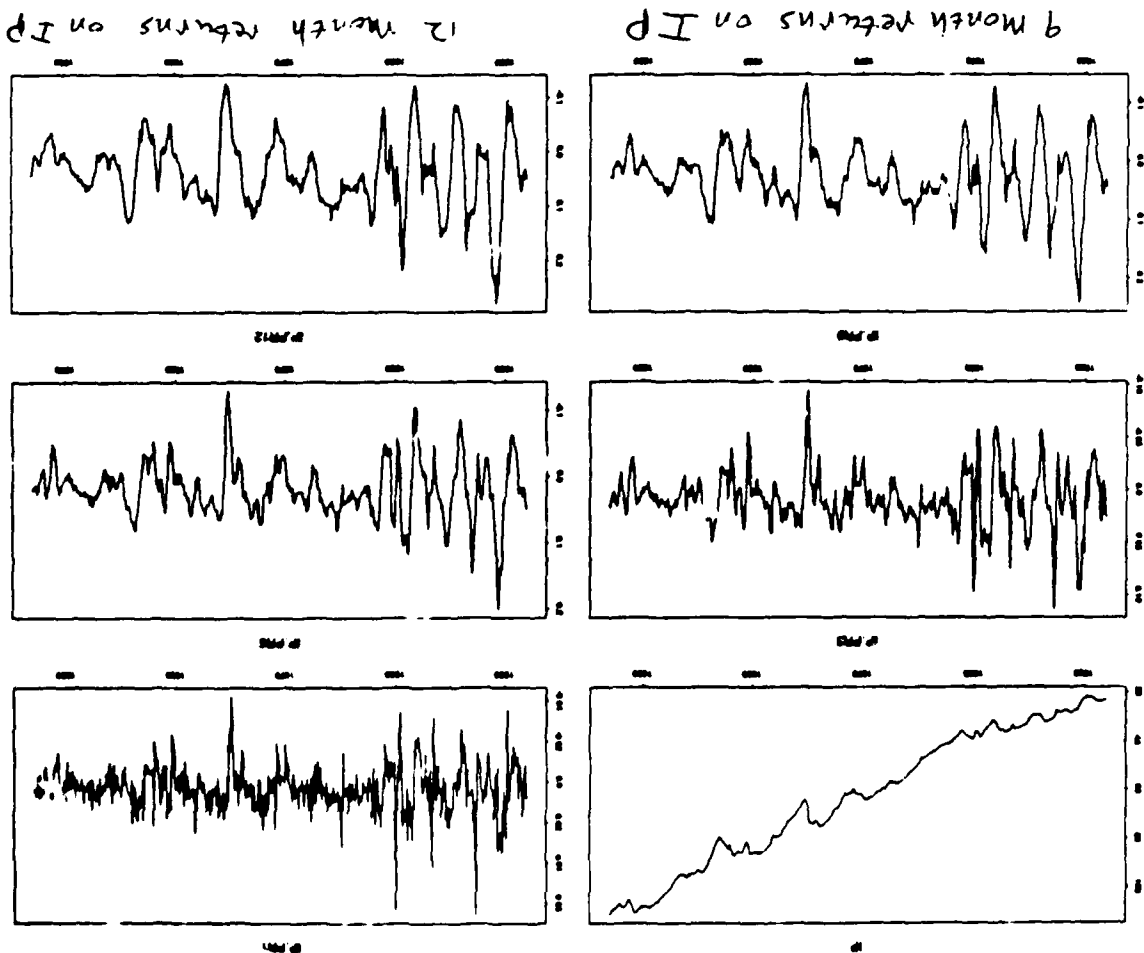


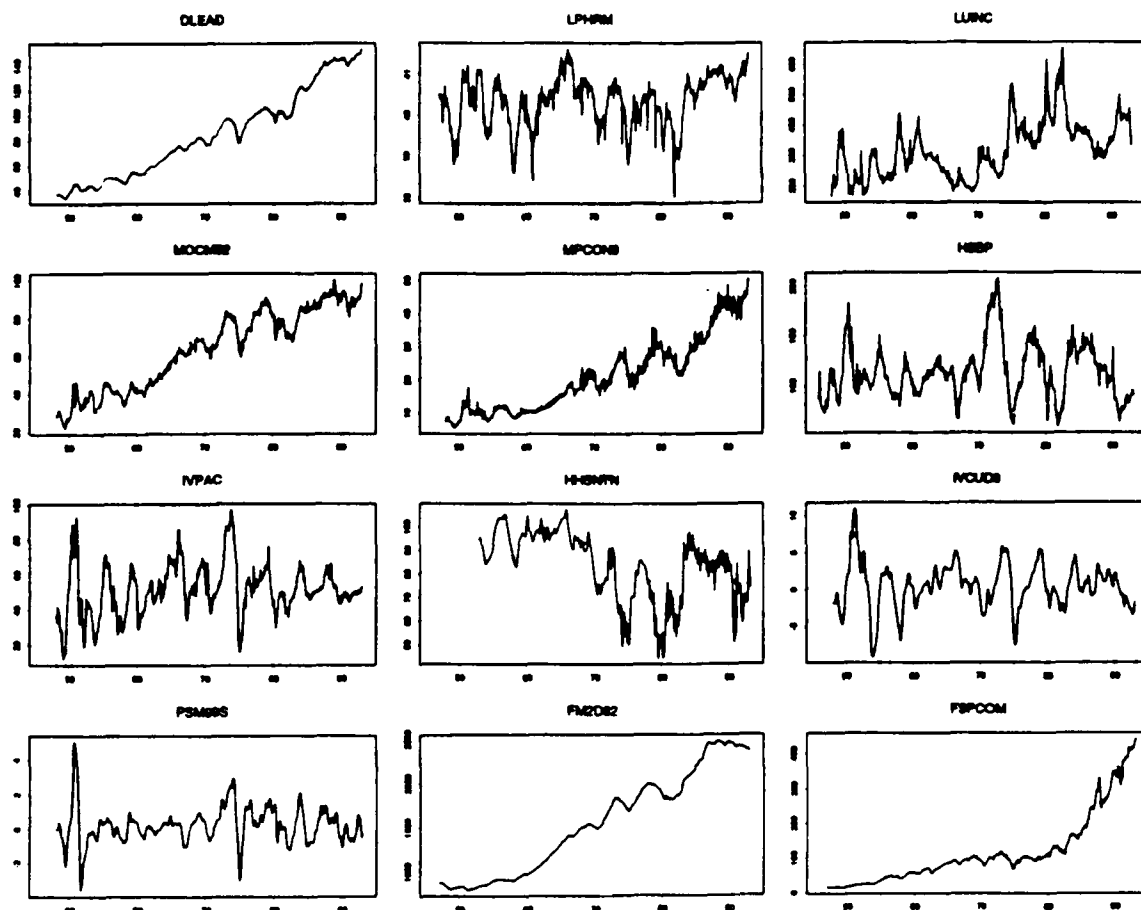
Fig. 1.1 Forecast distributions and actual values of percentage changes in real GNP, three horizons, 1968:IV-1990:I



Source: Zarnowitz & Braun 1993

Note the poor quality of professional economists' forecasts.





The Index of Leading Indicators and its Components.

General Issues regarding Predicting Industrial Production

* Poor Performance in GNP Forecasting by Professional Economists

* Convincing evidence for:

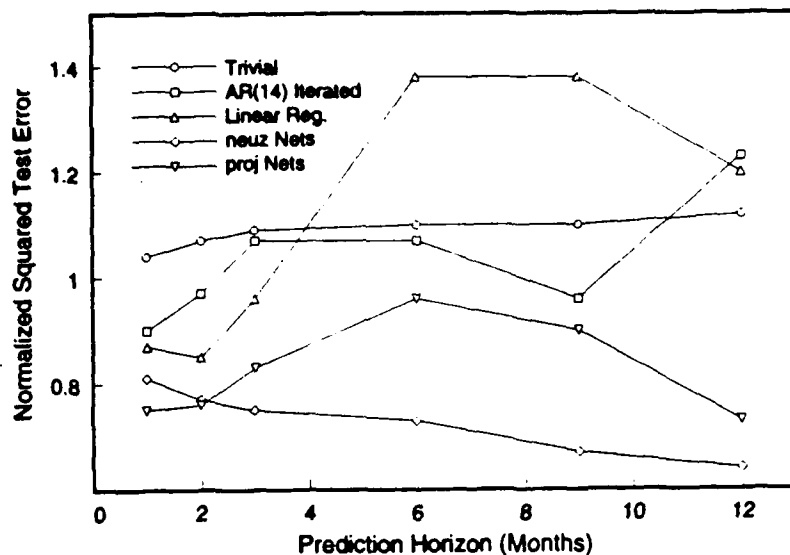
non-normality in I.P.
non-stationarity in I.P.

* Can we:

find convincing evidence for
non-stationarity in I.P.?
linearity

find a best model given the
"Noise - Non-stationarity Trade-off"?

Neural Nets Out-Perform Linear Models



Note: Relative Performance of NNs improves w/ Forecast Horizon $\Rightarrow$ Increasing Nonlinearity w/ Horizon

~~| | |
|-------------------------------|--------|
| Training Set Squared Error | 0.7632 |
| One Month Prediction Error | 0.9036 |
| Two Month Prediction Error | 0.9733 |
| Three Month Prediction Error | 1.071 |
| Six Month Prediction Error | 1.072 |
| Nine Month Prediction Error | 0.9599 |
| Twelve Month Prediction Error | 1.229 |~~

Table 2: Results of univariate AR(14) Model for IP.R1, one month rates of return for IP. Training set error and error for iterated predictions are shown.

Components of Leading Index

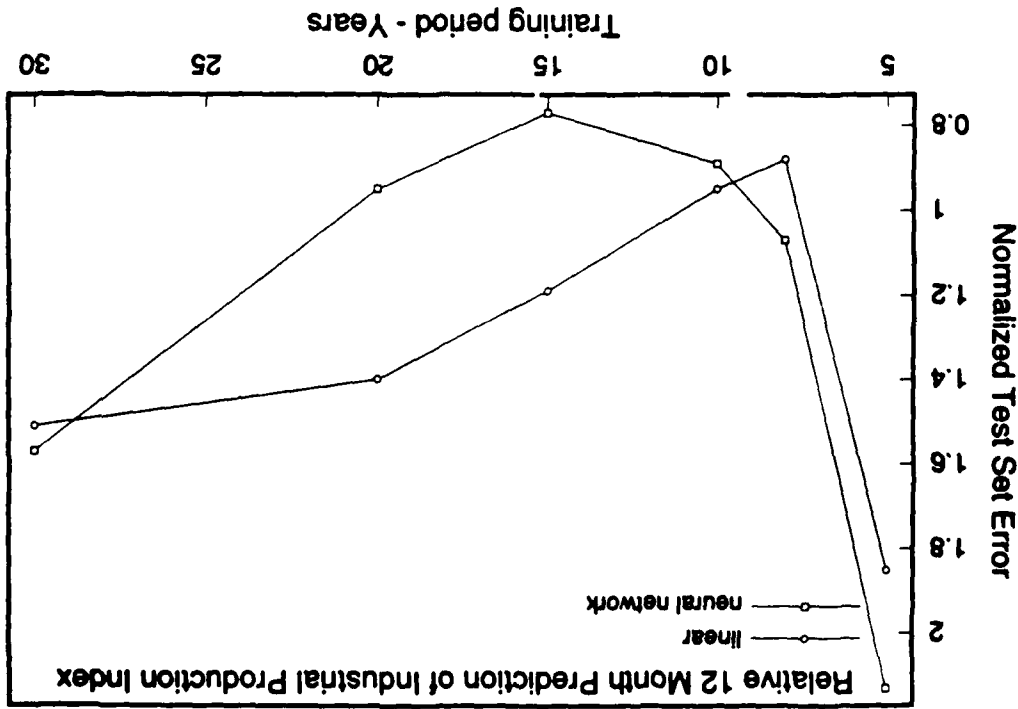
Symbol	Description
DLEAD	COMPOSITE INDEX OF 11 LEADING INDICATORS (82=100, SA)
LFPHM	AVG. WEEKLY HRS. OF PRODUCTION WKS.: MANUFACTURING (SA)
LUTNC	AVG. WKLY INITIAL CLAIMS STATE UNEMPLOY. INS.: EXC P.RICO (THOUS.SA)
MOC482	2 MFG NEW ORDERS: CONSUMER GOODS & MATERIAL IN 828 (BIL.SA)
MPCON8	CONTRACTS & ORDERS FOR PLANT & EQUIPMENT IN 828 (BIL.SA)
HSBP	HOUSING AUTHORIZED: INDEX OF NEW PRIV HOUSING UNITS (1967=100, SA)
IVPAC	VENDOR PERFORMANCE: OF CO'S REPORTING SLOWER DELIVERIES (NSA)
HHNSTN	U. OF MICH. INDEX OF CONSUMER EXPECTATIONS (BCD-83)
IVCUD8	CHANGE MFG UNFUL ORDER DUR. GOODS, SMOOTHED (BILL.828) (BCD-92)
P3M998	CHANGE IN SENSITIVE MATERIALS PRICES SMOOTHED DATA (BCD99)
FM2D82	MONEY STOCK: M-2 IN 1967 (BIL.SA) (BCD 106)
FSPCOM	S&P'S COMMON STOCK PRICE INDEX: COMPOSITE (1941-43=10)

Table 3: Symbols and descriptions for the Index of Leading Indicators and its 11 current components.

~~| AR Order (AR) | Training Set | Test Set |
|---------------|-----------------|--------------|
| 3 | 1950 to 1975.92 | 1980 to 1990 |
| Component | Training Error | Test Error |
| DLEAD.PRI | 0.962 | 0.7879 |
| IP.PRI | 0.7169 | 0.9481 |~~

Table 4: Results of bivariate AR Model for one month rates of return for DLEAD and IP.

Evidence for Neutrality the "Noise - Neutrality Trade-Off"

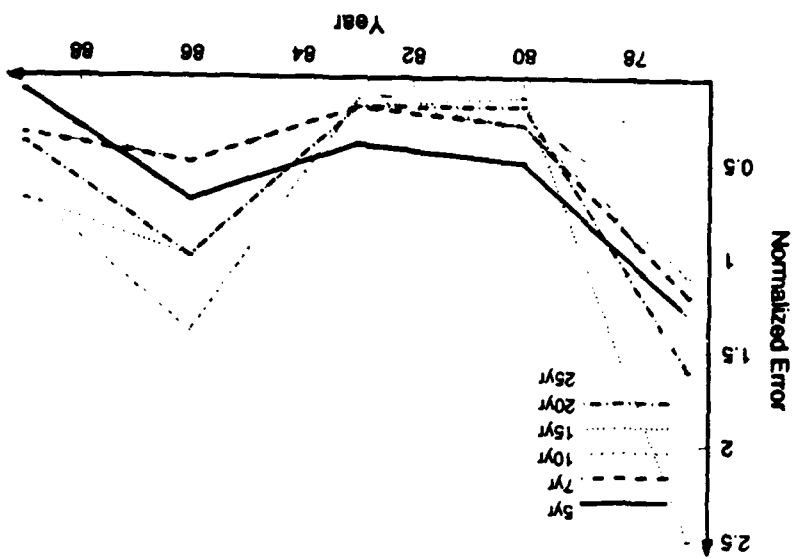


References on Combinations of Forecasts

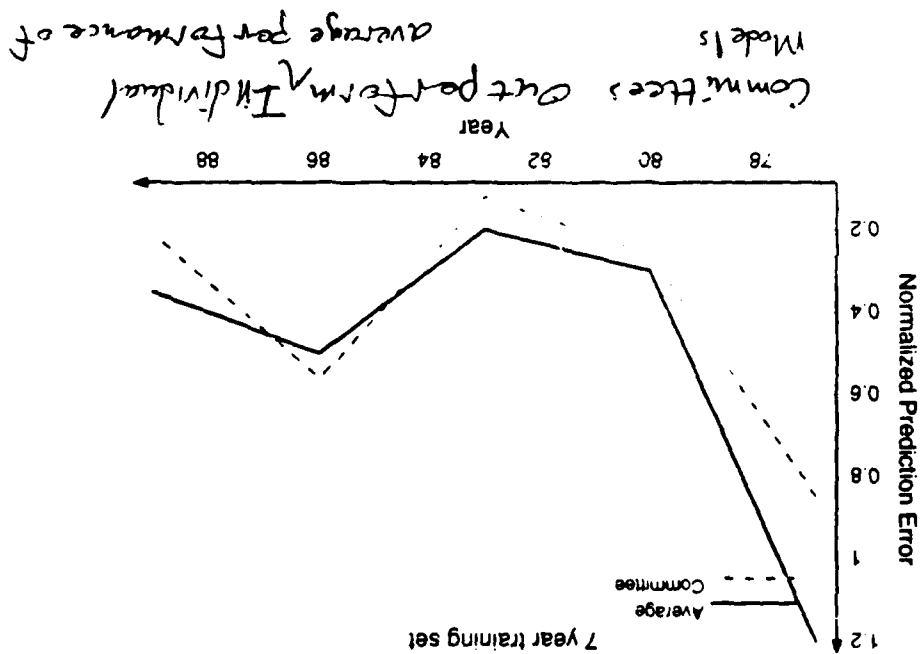
- Bates & Granger 1969
"The Combination of Forecasts"
Op. Res. Q 20 451-468.
- Newbold & Granger 1974
"Experience With ..."
J. Roy. Stat. Soc. A132, 131-146
- Winkler & Makridakis 1983
"The Combination of Forecasts"
J. Roy. Stat. Soc. A146, 150-157

Special issues of:
Journal of Forecasting 1989
Int'l. J. of Forecasting 1989

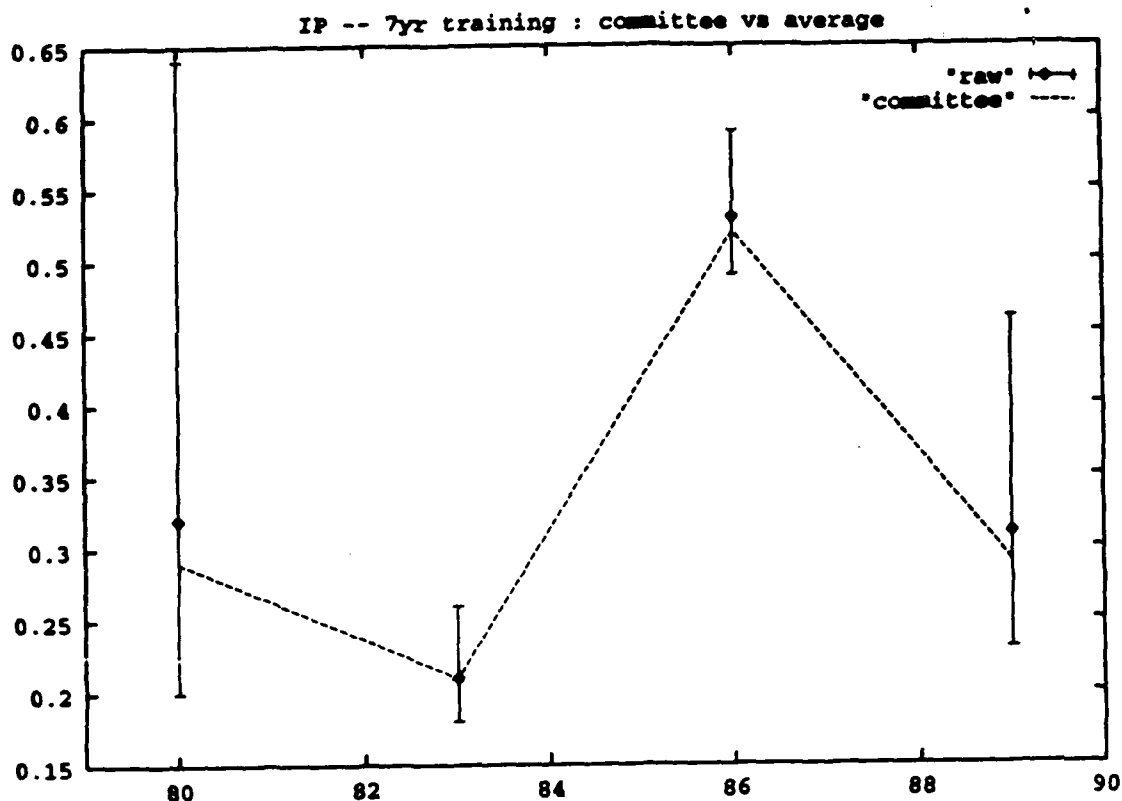
Book: Granger & Newbold 1986
Forecasting Economic Time Series, Ch. 9.
Academic Press



2025
10.12

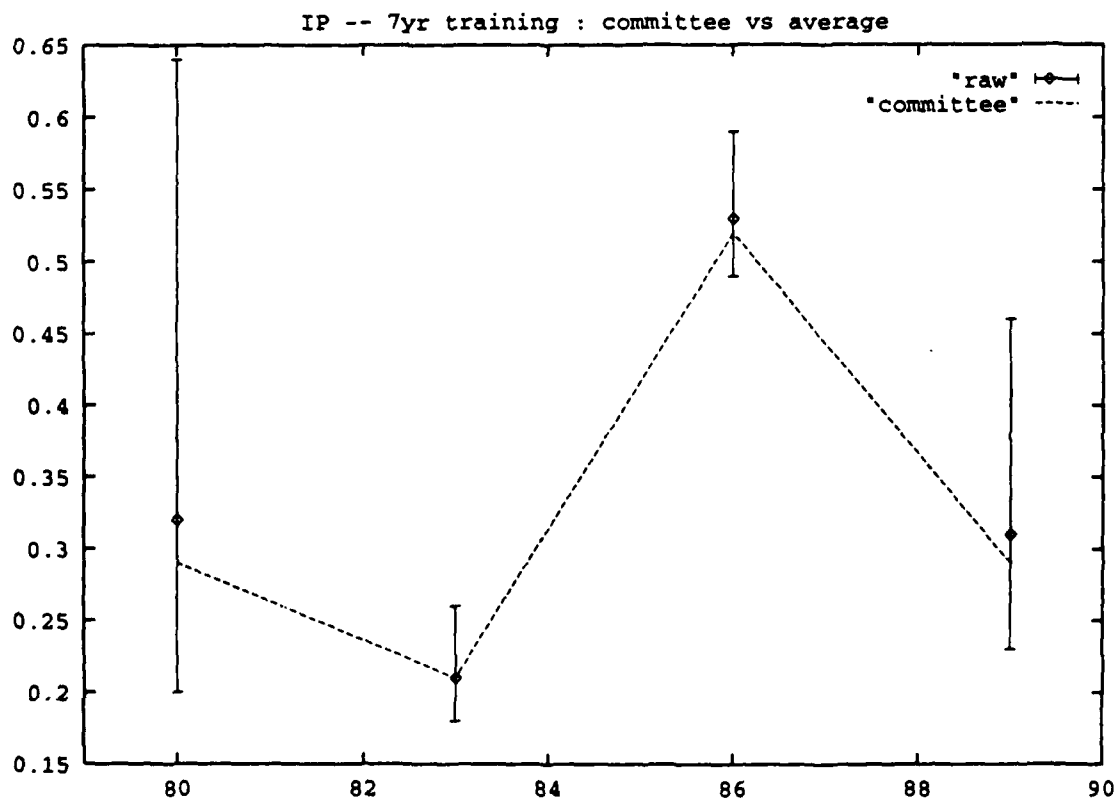


Committee's outperform individual models
average performance of



Performance of Committees is better on average and more consistent than individual models.

IP45 - new



Overview of combining models

Woody BunTINE
RIACS / NASA Ames Res. (Sr.)

Summary of 'Wolke & Carter '88.

(undergraduate thesis at Sydney)

method averaging multiple decision

trees: generate m trees by

perturbing the splitting rule then

combine their predictions

results

□ averaging better than voting

□ more models (larger m) better

initially, but with m too large it

can degrade (e.g. > 20)

(presumably due to other generation method)

□ averaging produces much better

class probability estimates (measured by

MSE / half-Brier score), whereas

accuracy improvement sometimes

only marginal (2%)

Basic formulation of learning

model M with real-valued parameters θ

cost function, $\text{cost}(M, \theta)$, usually

observed-error (M, θ) + complexity (M, θ)

choose (M, θ) to minimize cost

- complexity can be regularizer,
-log(Bayesian prior), encoding cost,
- what if model family is poor?
different model may match problem better, so give improved learning
- what if you have small data set
many models will be near minimum,
so seems unfair to pick just one
e.g. Medical investments

Mixture modelling

$$p(\text{resp. } | M, \theta, \text{input}) = \sum_c p(c | M, \theta, \text{input}) \cdot p(\text{resp. } | c, M, \theta)$$

giving fn. experts predicting

□ i.e. extend the model to be a mixture of experts ←

□ averaging is part of the network
so occurs inside the cost function

□ Theory says if the mixture is a better model of problem (e.g. Abowhan's speech task), then asymptotic mixture should be faster

(i.e. this is a very loose statement!
i.e. "better" model needs less parameters)

Model Averaging

pick multiple models $\left. \begin{matrix} M_1, \theta_1 \\ M_2, \theta_2 \\ \dots \\ M_m, \theta_m \end{matrix} \right\}$

chosen somehow with the basic cost minimization scheme then average their results.

$$n(\text{resp} | \text{Data}_{\text{input}}) = \sum_{i=1}^m p(\text{resp} | M_i, \theta_i, \text{input}) w_i$$

□ averaging occurs outside the cost function

□ rationale is we aren't convinced that global cost minimum gives absolute "best" model since many others are almost as good

□ should do better for small samples and converge to maximum a posteriori

Bayesian Averaging

(rationale for model averaging)

$p(\text{new-response} | \text{Data, new-input})$

$$= \int_{M, \theta} p(\text{new-response} | M, \theta, \text{new-input}) p(M, \theta | \text{data})$$

model prediction model quality

$$\approx \sum_{(M_i, \theta_i)} p(\text{new-response} | M_i, \theta_i, \text{new-input}) w_i$$

weights for sampling scheme

ie. use Monte Carlo to approximate

the integral with finite sample of models

$$(M_1, \theta_1), \dots, (M_m, \theta_m)$$

e.g. Gibbs importance sampling, bootstrap, some "closed" form approximations

e.g. Neal, BUGS (Spiegelhalter et al.), Buntine '94, etc.

DISCUSSION

GEM
ensemble methods
committees
C&W

} model averaging
ie. better on small samples!

hierarchical mixture
of experts

} mixture modelling

boosting

? mixture modelling + regularization

since each weak classifier
trained on small subset
of training data
ensemble method on small samples

Computationally expensive
Champion for improving speed
of training by retraining
on parts of data

Inferring a function

vs.

inferring an inference algorithm

David Wolpert
SFI, TXN

OVERVIEW

- 1) $\exists$ millions of games to play
w partitions of the training set.
- 2) Alas, can't prove nuthin'.
- 3) Theory suggests thought -
that inferring an inference algorithm
is highly analogous to inferring an
i/o function.
- 4) Exploit this analogy.

INFERRING INFERENCE ALGORITHMS

- 1) Model selection w/o data partitioning
(E.g., AIC, GPE, ML-II)
- 2) Model combining w/o data partitioning
(E.g., Perrone, Newlan, Buntine)
- 3) Model selection w/ data partitioning
(cross-validation, stratification)
- 4) Model combining w/ data partitioning
Stacking: Breiman, Friedman, Waltz,...

Tons to do w/ (3) & (4).

For example...

STRONG CROSS-VALIDATION

Illustrate for noise-free case:

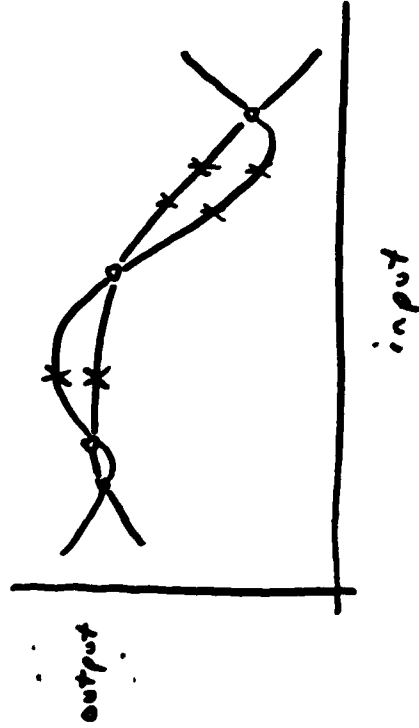


O = actual training
set element

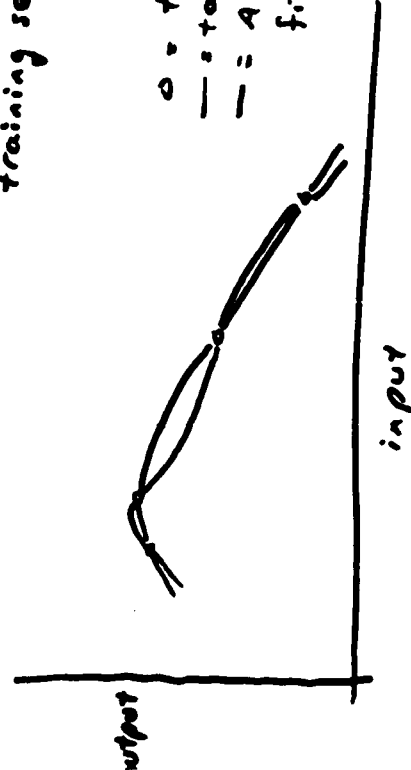
x = "alternative" training
set element

Would like small x -validation error
not only within the O 's, but within
any set of x 's lying on f .

Problem: Don't know f . But do know "proxies"...



○ = original training set element
 — = generalizer A's fit
 x = " " alternative training set element
 — = generalizer B's fit
 x = " " alternative training set element



○ = training set
 — = target function
 — = A generalizer's fit

1) No credit for memorization
 (can affect "learning curve" theoretical results drastically.)

2) Life often is not iid between testing & training.

3) The whole notion of "generalize" to new examples.

So it's worth considering off-
 test training error (i.e. test set disjoint from training set) as well as conventional test error (i.e. test set can overlap training set).

THEORY; you can't prove anything

- f is a target function, and m the size of L .
- $E_i(.)$ refers to an expectation value using generalizer i .

For any two generalizers, independent of the noise and sampling distribution,

i) Averaged over all f ,

$$E_1(E | f, m) - E_2(E | f, m) = 0;$$

ii) Averaged over all f , for any L ,

$$E_1(E | f, L) - E_2(E | f, L) = 0;$$

iii) Averaged over all $P(f)$,

$$E_1(E | m) - E_2(E | m) = 0;$$

iv) Averaged over all $P(f)$, for any L ,

$$E_1(E | L) - E_2(E | L) = 0.$$

THEORY; gotta make an assumption

1)

$$P(E | m) = \sum_{h,f} P(h | L) P(f | L) M_E(h, f)$$

For no noise, an inner product.

So must guess $P(f | L)$. Empirical Bayes then says ... cross-validate!

2)

$$P(E | s, L) = \sum_{h,f} P(h | L) P(f | L) N_{s,E,L}(h, f)$$

So “over-training” need have nothing to do with “training on the noise”; it can occur when there *is* no noise.

META-GENERALIZATION

A new event space. Define X' from X and Y :

$x' \in X'$ is an ordered set

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m); q\}.$$

Just as X and Y gave

f 's (mappings from X to Y);

h 's (mappings from X to Y);

L 's (sets of X - Y pairs)

obeying $P(h, f | L) = P(h | L) \times P(f | L)$,

X' and Y give

d 's (mappings from X' to Y);

g 's (mappings from X' to Y);

ω 's (sets of X' - Y pairs)

obeying $P(d, g | \omega) = P(g | \omega) \times P(d | \omega)$.

META-GENERALIZATION - continued

If $P(g | \omega)$ is the rule "choose the g best fitting ω "

- analogous to "choose the h best fitting L ".

then you have leave-one-out cross-validation.

$$P(E | m) = \sum_{g,d} P(g | \omega) P(d | \omega) M'_E(g, d)$$

$$P(E | s, \omega) = \sum_{d,g} P(g | \omega) P(d | \omega) N'_{s,E,\omega}(g, d)$$

So:

Over-cross-validating, just like over-training;

Regularized cross-validation;

Combining generalizers - stacking - just like combining hypotheses.

Theory gives insight, not answers.

A Bayesian Perspective on Committees

Hans H. Thodberg, Danish Meat Res. Inst.

email: thodberg@mm.meatres.dk

Summary of Nips'93 poster

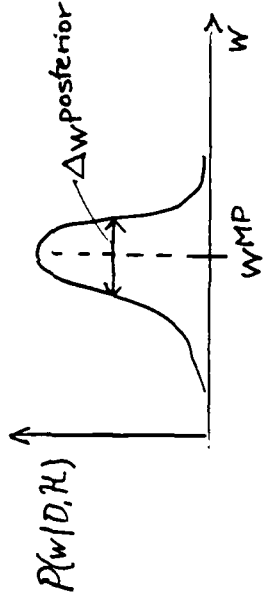
MacKay's Bayesian framework for Backpropagation is applied and extended.

- Adjusts weight decay parameters on-the-fly
- No test set is needed
- The evidence for a model = Quality measure.
High evidence implies good generalisation
- The evidence is an efficient stop criterion for pruning
- The evidence of a committee is computed
- Bayesian error bars on predictions are computed
- Real-World Application to analytic chemistry:
The determination of fat in minced meat by near-infrared spectroscopy.

Bayes Gives the Answer

$$P(\mathcal{H}|D) = \frac{\overbrace{P(D|\mathcal{H})P(\mathcal{H})}^{\text{the evidence for } \mathcal{H}}}{P(D)}$$

$$P(w|D, \mathcal{H}) = \frac{P(D|\mathcal{H}, w)P(w|\mathcal{H})}{P(D|\mathcal{H})}$$



$$P(D|\mathcal{H}) = \int P(D|\mathcal{H}, w)P(w|\mathcal{H})dw$$

$$P(w|\mathcal{H}) = 1/\Delta w^{\text{prior}}$$

$$Ev(\mathcal{H}, \equiv P(D|\mathcal{H}) = P(D|\mathcal{H}, w^{\text{MP}}) \frac{\Delta w^{\text{posterior}}}{\Delta w^{\text{prior}}}$$

Evidence = Likelihood $\times$ OckhamFactor

ModelQuality = DataFit $\times$ Simplicity

The Four Levels of Inference

Level 1 Make predictions including error bars for new input data.

Level 2 Estimate the weight parameters and their uncertainties.

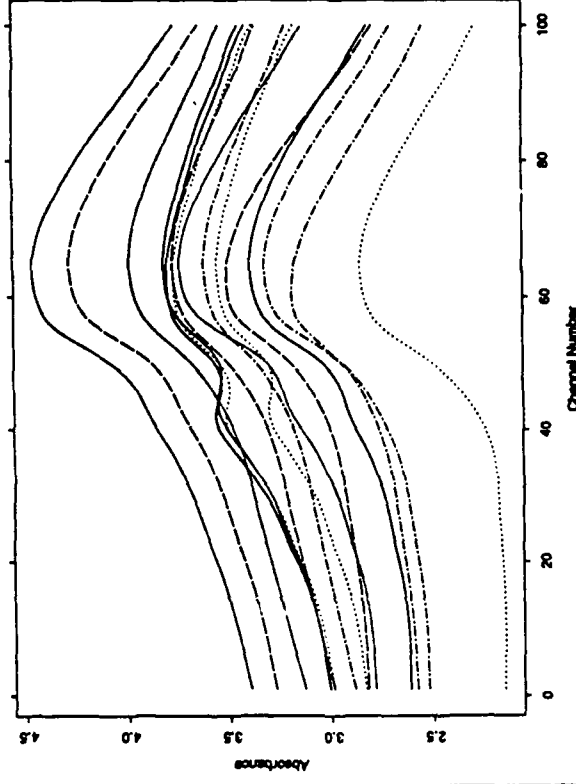
Level 3 Estimate the scale parameters and their uncertainties

Level 4 Select the w -minimum and the network architecture. Optionally select a committee to reflect the uncertainty on this level.

Application to Spectroscopic Data

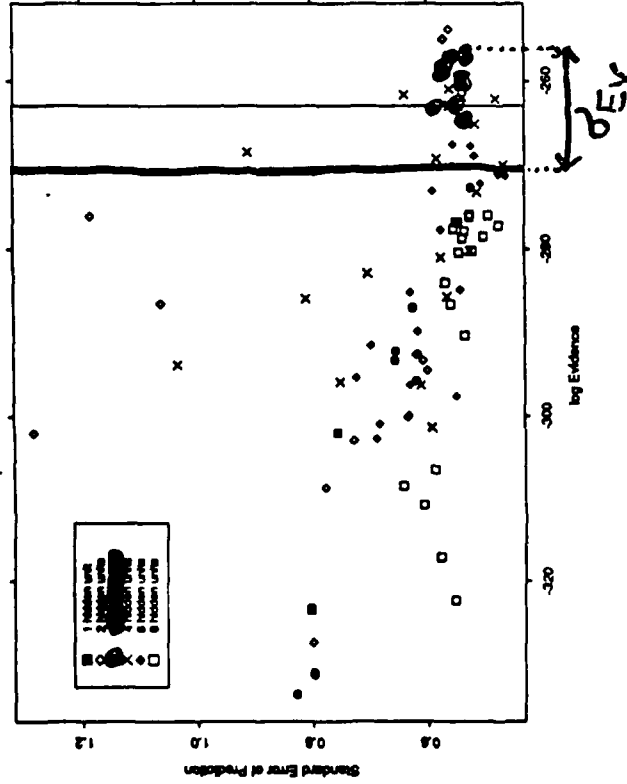
A Tecator near-infrared spectrometer is used to measure the light transmitted through a sample of ground meat. The purpose is to determine the fat content, which varies between 5 and 50%.

NIR Spectroscopy



Fifteen NIR-transmission spectra

Spectroscopy



The test error as a function of the log evidence for networks trained on the spectroscopic data. Networks with 2 and 3 hidden units attain the highest evidence. The 20 networks with highest evidence are delimited by the vertical line and have on average $SEP=0.55$.

The Evidence for a Committee of Networks

Train several networks with different numbers of hidden units and different initial weights.

Select a committee consisting of the networks with the best evidence within an estimated uncertainty

- The average of the committee gives a better generalisation than the average network in the committee.
- The degree of dissent within the committee is used to compute the uncertainty of the prediction.

N_c is the number of different solutions in the committee (same architecture).

$Ock(w)$ is N_c times smaller:

$$\log Ev(C) = \log N_c + \log Ev(\mathcal{H})$$

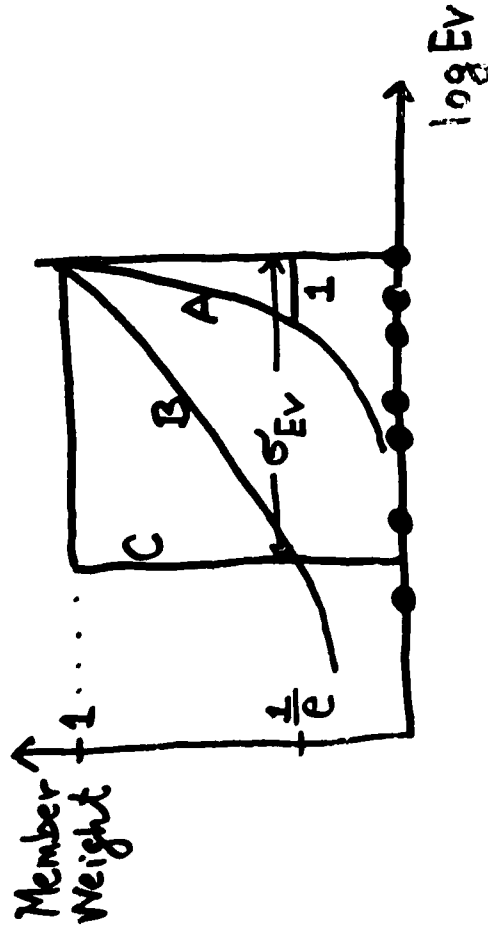
How to weight the members of the committee?

→ according to the evidence

$$Ev(x) = p(D|x)$$

However...

$$\log Ev_{GaussApp}(x) = \log Ev(x) + \Delta Ev \quad \leftarrow N(0, \sigma_{Ev}^2)$$



I use Member Weight C as an approximation to B

The committee prediction

$$y(x) = \frac{1}{N_c} \sum_{j=1}^{N_c} y(x, w_{MP}^j)$$

The committee members disagree to a certain extent and this committee uncertainty (CU) gives the following contribution to the prediction variance:

$$\sigma_{cu}(x)^2 = \frac{1}{N_c - 1} \sum_{j=1}^{N_c} (y(x) - y(x, w_{MP}^j))^2$$

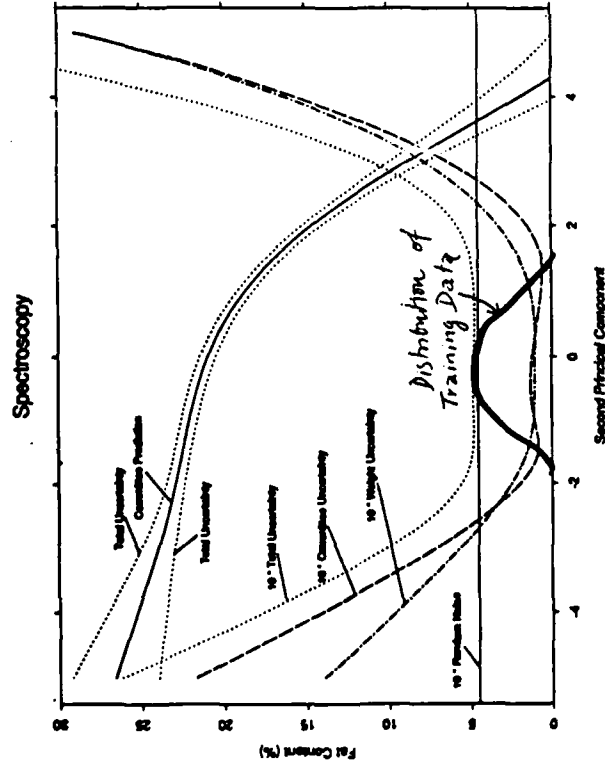
The total prediction variance is

$$\sigma_{total}(x)^2 = \sigma_v^2 + \sigma_{wo}(x)^2 + \sigma_{cu}(x)^2$$

See the figure!

Conclusions

- Bayes gives a rational account for committees
(Ockham does not! — "simple" committees are not "simple")
- Realistic Error Bars requires Committee Uncertainty term
- Principle for Active Learning:
Ask if the ~~experts~~ experts disagree



Prediction of the fat content as a function of the second principal component p_2 of the NIR spectrum. 95% of the training data has $|p_2| < 2$. The total standard error bars are indicated by a "1 sigma" band (dotted lines). The total standard errors and the standard errors of its contributions are shown separately, multiplied by a factor of 10.

References

- H.H.Thodberg, "A Review of Bayesian Neural Networks with an Application to Near Infrared Spectroscopy" and "A Bayesian Approach to Pruning of Neural Networks" submitted to *IEEE Trans. of Neural Networks* 1993. (in [thodberg.ace-of-bayes.ps.Z](#) in the [neuroprose](#) electronic archive at Ohio)
- D.J.C.MacKay, "A Practical Bayesian Framework for Backpropagation Networks" *Neural Comp.* 4 (1992) 448-472.
- H.H.Thodberg, "Improving Generalization of Neural Networks through Pruning", *Int.J.Neural Systems* 1 (1991) 317-325.
- C.Bishop, "Exact Calculation of the Hessian Matrix for the Multilayer Perceptron" *Neural Comp.* 4 (1992) 494-501.
- C.Borggaard and H.H.Thodberg, "Optimal Minimal Neural Interpretation of Spectra", *Analytic Chemistry* 64 (1992) 545-551.

OUTLINE

- Optimal Linear Combinations of Neural Networks.
 - Motivation.
 - Definition.
 - Combination weights.
- Benefits of Combining:
 - For well-trained networks.
 - For poorly trained networks.
- Ill Effects of Collinearity:
 - Computational ill effects.
 - Statistical ill effects.
- Concluding Remarks.

MERITS OF COMBINING NEURAL NETWORKS POTENTIAL BENEFITS AND RISKS

SHERIF HASHEM¹

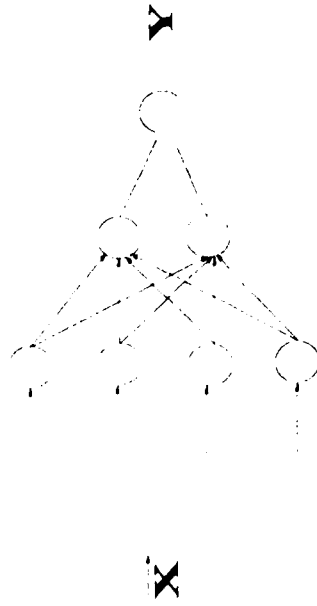
SCHOOL OF INDUSTRIAL ENGINEERING
PURDUE UNIVERSITY

DECEMBER 4, 1993

¹Supported by PFI Research Grant 6001627 from Purdue University, West Lafayette, IN

OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS

MOTIVATION



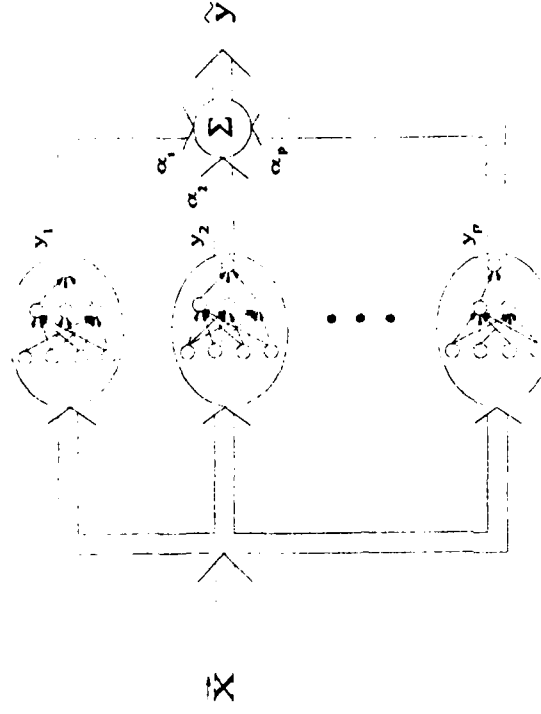
Neural Network Model

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

Problem: Given a (process) data set, construct a Neural Network based model that "closely" approximates the underlying process.

OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS

DEFINITION



$$\hat{Y} = \sum_{i=1}^p \alpha_i Y_i = \vec{\alpha}' \vec{Y}$$

- **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.

$$\text{MSE} = E_{\vec{X}}(\tau(\vec{X}) - \tilde{Y})^2.$$

The MSE-optimal weights:

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

where

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

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

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

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

OTHER FORMS OF MSE-OLC

A. Unconstrained MSE-OLC with a constant term.

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

B. Constrained MSE-OLC with a constant term.

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

C. Constrained MSE-OLC without a constant term.

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

D. Convex MSE-OLCs:

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

EXAMPLE 1

- **Problem:** Consider approximating

$$r(X) = 0.02(12 + 3X - 3.5X^2 + 7.2X^3)(1 + \cos 4\pi X)(1 + 0.8 \sin 3\pi X)$$

where $X \in [0, 1]$.

- **NN model:**

- *Topology:* Three 1-5-5-1 & three 1-10-1 NNs
- *Training algorithm:* Error backpropagation
- *Training data:* 200 uniformly distributed points $(x, r(x))$
- *MSE-OLC fitting data:* same 200 points

- **Resultant "true" MSE:**

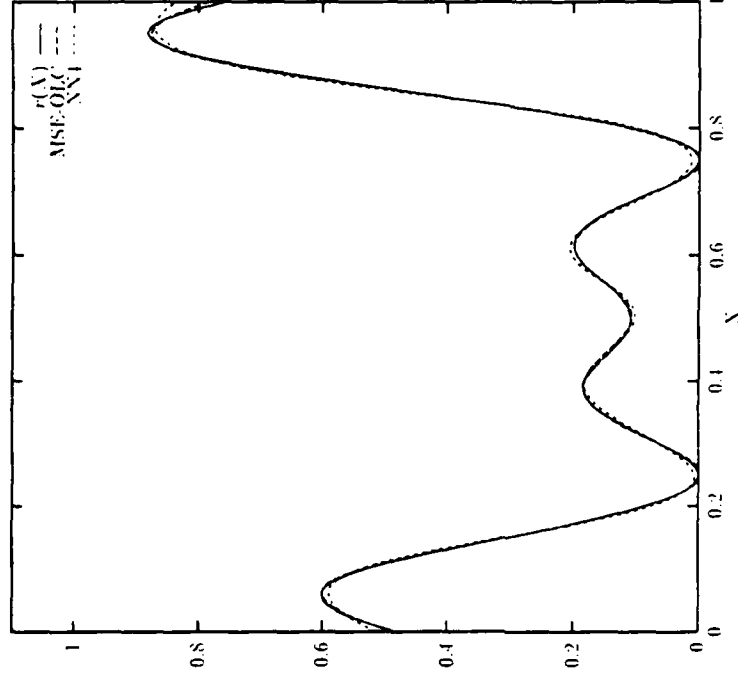
Best NN (NN₄): 0.000137.

Simple averaging: 0.000396.

MSE-OLC: 0.000017;

88 % less than NN4; 96 % less than simple averaging.

EXAMPLE 1 (Cont.)



¹Computed on the (true) known function $r(X)$

BENEFITS OF COMBINING FOR WELL-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 5000 iterations.
- *Training data:* 10 uniformly distributed data points.
- *MSE-OLC fitting data:* same 10 points.

- **Resultant “true” MSE:**

Best NN (NN6): 0.044.

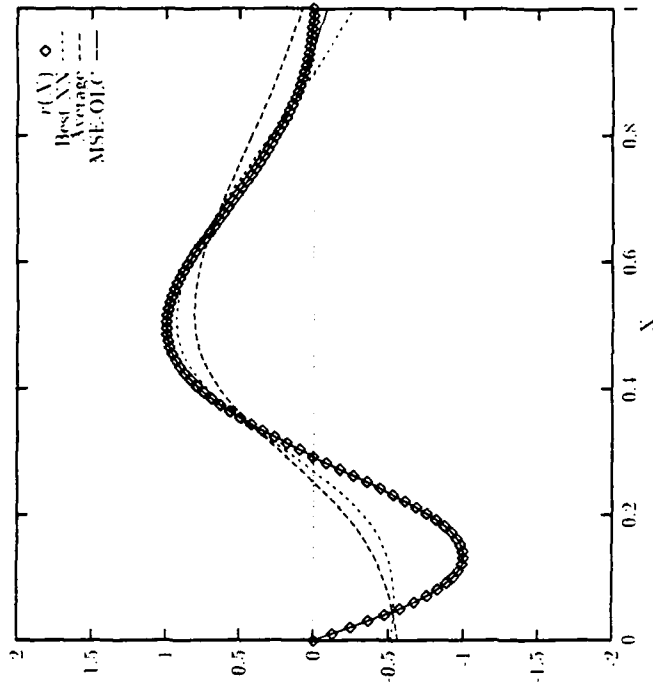
Simple averaging: 0.072.

MSE-OLC: 0.00020.

99+ % less than NN6 or simple averaging.

BENEFITS OF COMBINING FOR WELL-TRAINED NETWORKS

EXAMPLE 2 (Cont.)



BENEFITS OF COMBINING FOR POORLY TRAINED NETWORKS

EXAMPLE 3

- **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.

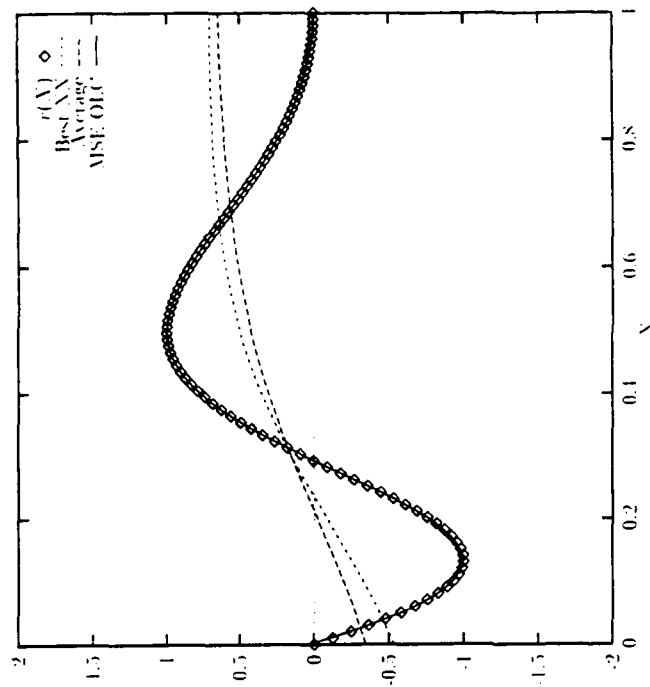
Simple averaging: 0.241.

MSE-OLC: 0.000060.

99 + % less than NN6 or simple averaging.

BENEFITS OF COMBINING FOR POORLY TRAINED NETWORKS

EXAMPLE 3 (Cont.)



ILL EFFECTS OF COLLINEARITY

MSE-OLCs:

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

• Unconstrained MSE-OLC Weights:

$$\tilde{\alpha}^* = \Phi^{-1} \tilde{\phi},$$

where

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

Constrained MSE-OLC Weights:

$$\tilde{\alpha}^* = \Omega^{-1} \tilde{\Gamma} / (\tilde{\Gamma}' \Omega^{-1} \tilde{\Gamma}),$$

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

Computational III Effects:

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

Statistical III Effects:

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

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.

REFERENCES

- R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*. 5:559-583, 1989.
- D. A. Belsley. *Conditioning Diagnostics: Collinearity and Weak Data in Regression*. John Wiley & Sons, New York, 1991.
- C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114-148, 1986.
- S. Hashem. *Optimal Linear Combinations of Neural Networks*. PhD thesis, School of Industrial Engineering, Purdue University, Dec. 1993.