A Generic Sensor Fusion Problem: Information and Computational Issues

Nageswara S.V. Rao
Center for Engineering Science Advanced Research
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6355
email: raons@ornl.gov

December 9-10, 1999
BMDO Workshop on
Advanced Algorithms for Data Fusion
ANSER, Crystal City, VA

This research is sponsored by
U.S. Department of Energy,
Office of Science, Engineering Research Program (1993-2000)
Office of Naval Research (1995-2000)
National Science Foundation (1991-1993)

Presentation Outline

1. Generic Sensor Fusion Problem

1.1 Formulation and Examples

1.2 Finite Sample Solutions

1.3 Performance Issues

1.4 Applications

2. Detection/Classification Problems

2.1 Distributed Detection Problem

2.2 Sample-Based Solutions

3. Is Fusion Easier ?

3.1 Isolation Fusers

3.2 Projective Fusers

4. Metafusers

4.1 Fusion of Fusers

4.2 Practical Paradigm

5. Performance Issues

5.1 Finite Sample Guarantees

5.2 Computational Issues

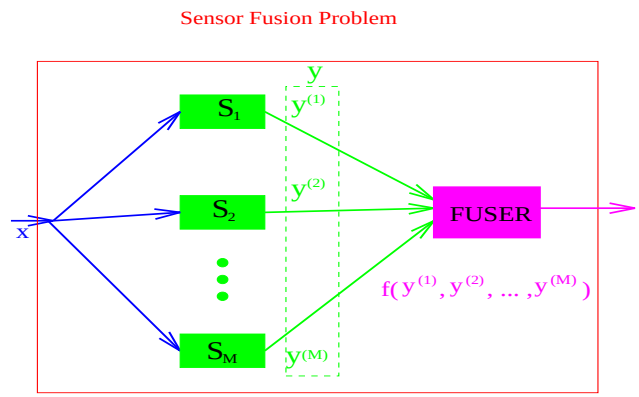
5.3 Practical Problems

Multiple Sensor System

Sensor System: $S_1, S_2, \ldots, S_M$

Sensor: S_i
$Y^{(i)} \in \Re$: output of S_i for input $X \in \Re$:

generated according to probability distribution $P_i(Y^{(i)}|X)$



Sensor Distributions: $P_i(.)$ could be
(i) known and in computationally conducive form
(ii) known but not in conducive form
(iii) unknown

Fusion Rule:
Given sensor outputs $(y^{(1)}, y^{(2)}, \ldots, y^{(M)})$
Compute input x

Fusion Function: Special case: $f : \Re^M \mapsto \Re$

Examples

Sensor could be hardware device, software module, or combination

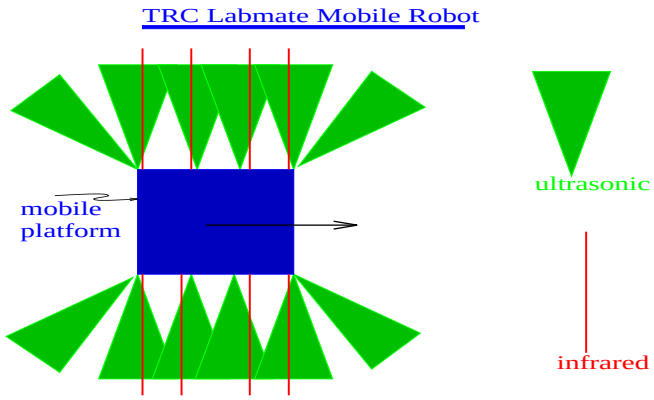
Example 1: Mobile Robot Navigation
$S_1, S_2, \ldots, S_N$: Array of ultrasonic and infrared sensors
Objective: Detect doors few inches wider/narrower than the robot
Sensor: Essentially hardware

Example 2: Function Estimation
S_i: Function estimator with unknown noise distribution
Objective: Combine predictors for more accurate prediction
Sensor: Entirely software

Example 3: Human DNA Analysis
S_i: Neural network/genetic algorithm to locate protein coding regions
Objective: Combine the modules to improve performance
Sensor: Entirely software

Example 4: Face Recognition
S_i: Camera, detection software, neural network recognizer for a class
Objective: Fast detection for a database of facial images
Sensor: Combination of hardware and software

Detection of Door



Array of ultrasonic/infrared sensors:

Why these sensors ?
Low cost, Low data content

Difficulties:

(i) Ultrasonic sensors are unreliable:
indicate objects when there are none;
give reasonable readings when directly facing flat objects

(ii) Infrared sensors:
Boolean detection in a direction;
occasional malfunctioning

Learning to Detect Door:
Using trials and providing yes/no information when a door is detected

Detection of Door

Ultrasonic Sensors:
$Y^{(1)}, Y^{(2)}, Y^{(3)}, Y^{(4)}$: normalized distance measurements

Infrared Sensors:
$Y^{(5)}, Y^{(6)}, Y^{(7)}, Y^{(8)}$: Boolean detection

Taining Data: 6 positive examples and 12 negative examples

Robot successfully detected passable doors

Practical Engineering Systems

Distributions:
(i) Very rarely known – often only a sample is given
(ii) Even when known not in computationally conducive form
(iii) Very hard to estimate - expertise in multiple areas needed

Sample: Typically small
– Asymptotic statsitical results do not offer much performance gurantees

Computation: Computation of fuser must be efficient
– Preferably low polynomial-time

Fusion Rule Estimation Problem

Expected error of fusion function f

$$I(f) = \int \Theta[X, f(Y)] dP_{Y|X}^{[1,M]} dP_X,$$

where
$P_{Y|X}^{[i,j]}$ is the joint distribution of $S_i, \ldots S_j$, and
$\Theta : \Re^2 \mapsto \Re$ is the cost function.

Best Fusion Function:
Let f^* minimize $I(f)$ over a family of functions $\mathcal{F}$.

Performance Parameters:
We compute a fusion rule $\hat{f}$:

- **Near-Optimality:** $I(\hat{f})$ must be *close* to $I(f^*)$
- **Computational Complexity:** $\hat{f}$ must be computable *fast*

Known Probabilities Case:

- If all distributions are known, f^* can be "solved" by optimization
- Computational problem is NP-hard even for simple systems

Fusion Rule Estimation

In our formulation:

- underlying probability distributions are **unknown**

- Only empirical data is known iid $(X_1, Y_1), (X_2, Y_2), \ldots, (X_l, Y_l)$

Result:
Only an approximation $\hat{f}$ to f^* is possible

PAC Criteria: Under Feasibility Conditions:
For sufficiently large sample,

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta$$

Note:

1. This is best type of guarantee possible under a finite sample

2. Such guarantees are not always possible if $\mathcal{F}$ is not finite

Classification/Detection problem

Special Case: Each S_i is a target detector or classifier:
$X \in \{0,1\}$, $Y^{(i)} \in \{0,1\}$, and

$$I\ (S_i) = \int [X \oplus Y^{(i)}] dP_{Y^{(i)},X}$$

corresponds to the probability of misclassification of S_i

Finite Sample Implications: Devroye (1982)
For any classifier C, any finite sample size l, there is a distribution such that:
(i) error of the Bayes classifier is 0;
(i) probability of misclassification of C $\geq 1/2 - \epsilon$,
for arbitrarily small $\epsilon > 0$.

Why fuse classifiers ?
S_i could be kernel rule, histogram, neural network, support vextor machine, boosted classier, or tree classifier
— there is no single best classifier for finite sample
— "fuser" can provide better finite sample guarantees,

Some Distributions are Known

In our formulation:

- Some underlying probability distributions are **unknown**
  Let $P_1(.), P_2(.), \ldots, P_N(.)$, $N \leq M$ be **unknown**
- Only empirical data is known $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$

Result:
Only an approximation $\hat{f}$ to f^* is possible

Decomposition:
Redefine cost function so that

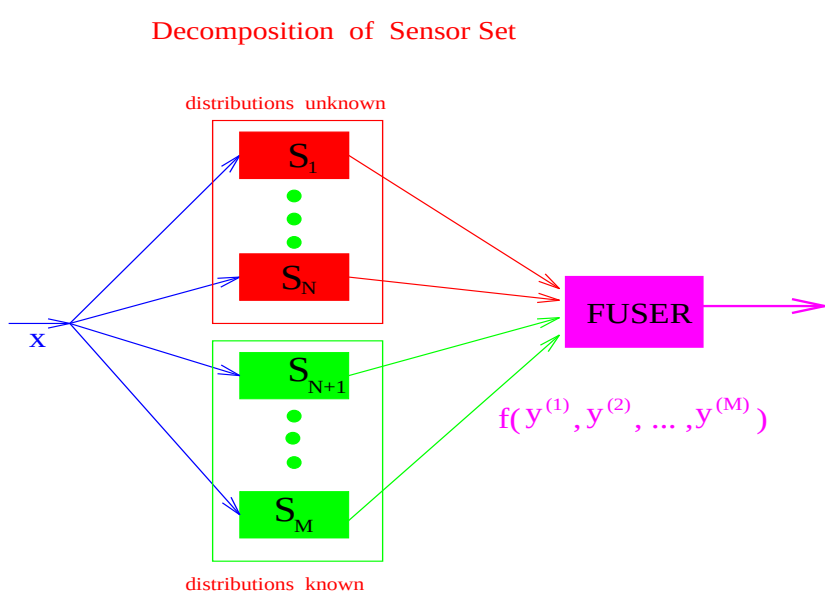
$$I(f) = \int C[X, f(Y))] dP_{Y|X}^{[1,N]} dP_X,$$

and

$$C[X, f(Y)] = \int \Theta[X, f(Y)] dP_{Y|X}^{[N+1,M]}$$

PAC Criteria: Under Feasibility Conditions:
For sufficiently large example,

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta$$



General Solution: Empirical Estimation

Expected square error: $\Theta[X, f(Y)] = [X - f(Y)]^2$

Vapnik's Method: Minimize empirical risk

$$I_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l}[X_i - f(Y_i)]^2$$

Consider:

$$I(f^*) = \min_{f \in \mathcal{F}}\{I(f)\}$$

$$I_{emp}(\hat{f}) = \min_{f \in \mathcal{F}}\{I_{emp}(f)\}$$

PAC Criteria: Under Feasibility Conditions:
For sufficiently large example, with probability $1 - \delta$ we have

$$I(\hat{f}) \leq I(f^*) + \epsilon$$

or equivalently

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta$$

Necessary and Sufficient Conditions
Infinite Hypothesis Space

Given l examples;
scale-sensitive dimension of $\mathcal{F}$ is d at precision $\epsilon/4$:

Bounded Loss:

$$\sup_{x,y}(x - f(y))^2 \leq 1$$

Performance Condition: Given a sample of size

$$\frac{5040}{\epsilon^2} \max\left\{d \ln^2 \frac{50d}{\epsilon}, \ln \frac{48}{\delta}\right\}$$

where $d = P_{\epsilon/4}$-dim $(\mathcal{F})$, we have

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta.$$

Example:
$f(y) = f_\alpha(y)$: Feedforward neural network with connection weight vector α.

Note:
1. Necessary condition was an open problem until now.
2. Made possible from a result from computational learning theory.
3. All previous results are sufficiency conditions, e. g. Vapnik's capacity, Pollard's dimension, metric entropy.

Feasibility Conditions - Part A

For family $\{A_\gamma\}_{\gamma\in\Gamma}$, $A_\gamma \subseteq A$, and
for a finite set $\{a_1, a_2, \ldots, a_n\} \subseteq A$

$$\Pi^A_{\{A_\gamma\}}(\{a_1, a_2, \ldots, a_n\}) = \{\{a_1, a_2, \ldots, a_n\} \cap A_\gamma\}_{\gamma\in\Gamma}$$

$$\Pi_{\{A_\gamma\}}(n) = \max_{a_1, a_2, \ldots, a_n} |\Pi_{\{A_\gamma\}}(\{a_1, a_2, \ldots, a_n\})|$$

Critical Identity:

$$\Pi_{\{A_\gamma\}}(n) = \begin{cases} 2^n & \text{if } n \leq h \\ < 1.5\frac{n^h}{h!} & \text{if } n > h \end{cases}$$

Vapnik-Chervonenkis Dimension of $\{A_\gamma\}$:

Largest size h of a set $\{a_1, a_2, \ldots, a_n\} \subseteq A$ that can be subdivided in all possible ways into two classes by means of sets A_γ.

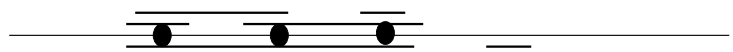
Formally,

$$VC - dim(\{A_\gamma\}) = \max_n \{\Pi_{\{a_1, a_2, \ldots, a_n\}}(n) = 2^n\}$$

Examples

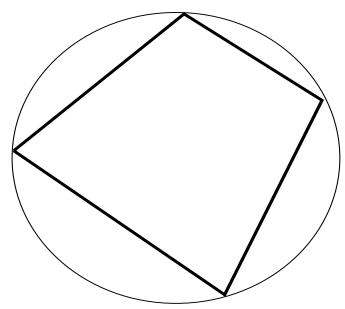
Example 1:
$\mathcal{A}$: Set of intervals of real line
$VC - dim(\mathcal{A}) = 2$

(a) entire set is shattered



(b) no interval can precisely include the two end points

Example 2:
$\mathcal{A}$: Set of all convex polygons with vertices on unit circle
$VC - dim(\mathcal{A}) = \infty$



Can create a polygon to include
any set of points

Example 3:
$\mathcal{A}$: Decision regions of single hidden-layer feedforward networks with threshold units with w weights
$VC - dim(\mathcal{A}) = O(w \log w)$

Capacity of $\{f_\alpha(x)\}_{\alpha\in\Lambda}$:

Capacity is largest VC-dim of indicator functions

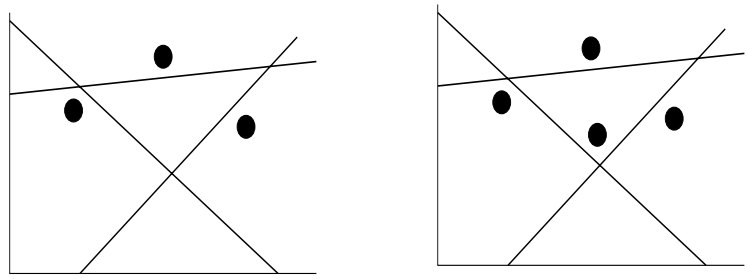
$$\{\Theta[f_\alpha(x)+\beta]\}_{\alpha\in\Lambda}$$

for $\beta \in \Re$, where

$$\Theta(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{f } z < 0 \end{cases}$$

Examples

Example 1:
$f(y, \alpha) = \sum_{i=1}^{d} \alpha_i y_i + \alpha_o$
where $y = (y_1, y_2, \ldots, y_d) \in \Re^d$ and $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_d) \in \Re^d$
Capacity of $\{f(y, \alpha)\}_{\alpha \in \Re^d}$ is $d + 1$



No hyperplane contains
precisely the two middle points

Example 2:
$f(y, \alpha)$: Sigmoidal feedforward neural network
Capacity of $\{f(y, \alpha)\}_{\alpha \in \Re^d}$ is finite

Feasibility Conditions: Infinite Hypothesis Space

Given n examples;
Capacity of $\mathcal{F}$ is h:

Bounded Loss:

$$\sup_{x,y}(x - f(y))^2 \leq \tau$$

Performance Condition:

$$P[I(\hat{f}) - I(f^*) \geq \epsilon] \leq \delta$$

where

$$\epsilon = 2\tau\kappa$$

and

$$\delta = 9\frac{(2n)^h}{h!}e^{-\kappa^2 n/4}$$

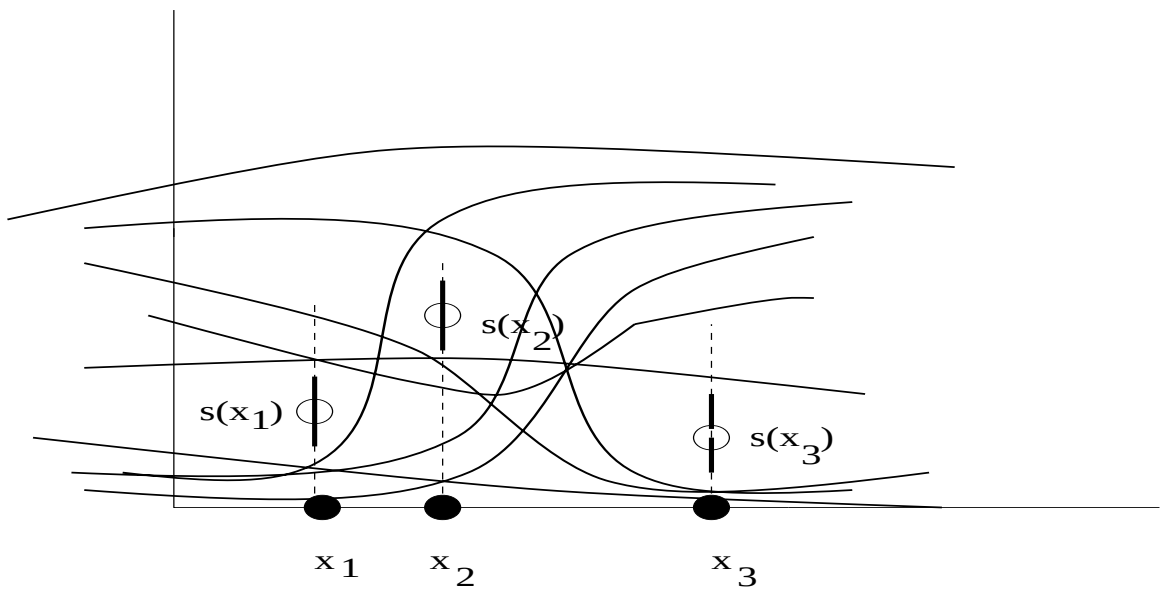
Example:
$f(y) = f_\alpha(y)$: Feedforward neural network with connection weight vector α.

Scale-Sensitive Dimension

Given:
$\mathcal{F}$: class of $[0,1]$-valued functions on D

P_ρ-shattering:
ρ: a positive real number
$\mathcal{F}$ *P_ρ-shatters* a set $A \subseteq D$ if there exists a function $s : A \mapsto [0,1]$ such that for every $E \subseteq A$ there exists some $f_E \in \mathcal{F}$ satisfying: for every $x \in A - E$, $f_E(x) \leq s(x) - \rho$, and for every $x \in E$, $f_E(x) \geq s(x) + \rho$.

P_ρ-dimension of $\mathcal{F}$: P_ρ-dim $(\mathcal{F})$:
maximal cardinality of a set $A \subseteq D$ that is P_ρ-shattered by $\mathcal{F}$.



Important Identity: $\mathcal{F}$: class of $[0,1]$-valued functions
P_ρ-dim $(\mathcal{F}) \leq 2\lceil \frac{1}{2\rho} \rceil \text{Capacity}(\mathcal{F})$

Note: Weakest characterization for the existence of finite sample results

<u>Outline of Proof:</u>

Step 1:
From Vapnik we have

$$P\left\{I(\hat{f}) - I(f^*) > \epsilon\right\} < P\left\{\sup_{g\in\mathcal{G}} |I_{emp}(f) - I(f)| > \epsilon/2\right\},$$

where $\mathcal{G} = \{(x - f(y))^2 : f \in \mathcal{F}\}$.

Step 2:
Let $\mathcal{G}_{2n} = \{g(X_1, Y_1), g(X_2, Y_2), \ldots, g(X_{2n}, Y_{2n}) | g \in \mathcal{G}\} \subseteq [0,1]^{2n}$, based on an iid sample of size $2n$.

From Alon *et al* , we have.

$$\begin{aligned} P\left\{\sup_{g\in\mathcal{G}} |I_{emp}(f) - I(f)| > \epsilon/2\right\} &\leq 24nE_{X^{2n}}[N(\epsilon/12, d_\infty^{2n}, \mathcal{G}_{2n})]e^{-\epsilon^2 n/144} \\ &\leq 48n\left(\frac{4608n}{\epsilon^2}\right)^{d\log_2(96en/(d\epsilon))} e^{-\epsilon^2 n/144} \end{aligned}$$

where $d = P_{\epsilon/4}$-dim $(\mathcal{F})$.

Step 3:
By equating the right hand side to δ, we obtain our sample size estimate.

Summary of Empirical Risk Minimization

Generalizations:

1. Quadratic cost can be generalized, e. g. Lipschitz
2. Boundedness condition can be replaced by relative boundedness
3. Structural risk minimization method

Computational Issues:

1. No systematic computational method exists to obtain $\hat{f}$
2. The problem is NP-hard for simple cases of $\mathcal{F}$, e.g. feedforward neural networks with Heaviside units (loading problem).
3. Linearly-separable systems: non-trivial example where $\hat{f}$ can be computed in polynomial-time using quadratic programming.

Need:

- Computationally efficient methods with finite sample guarantees

<u>Fusion Methods Studied:</u>

– Basic methods are known
– We developed: Finite sample results
Efficient computational algorithms

(i) **Sigmoidal neural networks with bounded weights**

– — Derived finite sample results
– — Computational problem is hard

(ii) **Vector space methods**

– — Potential functions
– — Kurkova's sigmoidal networks
– — Linear combinations

(iii) **Nadaraya-Watson estimator**

– — Derived finite sample results
– — Adapted to Haar wavelets - fast computation
– — Most effective in almost all applications

(iv) **Adapted nearest neighbor rules**

– — Adapted to Haar wavelets - fast computation
– — Derived finite sample results

There is no single "Best Method"
– For finite samples situation is very similar to classification

Sigmoid Feedfoward Networks

$\sigma : \Re \mapsto [-1, +1]$ called the *activation function*
input $r \in [-B, B]^d$, $b_j \in \Re^d$, $t_j \in \Re$.

$$f_w(r) = \sum_{j=1}^{l} a_j \sigma(b_j^T r + t_j)$$

where $w = (w_1, w_2, \ldots, w_{l(d+2)})$ consists
of a_1, a_2, ..., a_l, $b_{11}, b_{12}, \ldots, b_{1d}$, ..., $b_{l1}, \ldots b_{ld}$, and $t_1, t_2, \ldots, t_l$.

Sigmoid feedforward networks with *bounded weights*:
for $0 < \gamma < \infty$,

$$\mathcal{F}_{\mathcal{W}}^{\gamma} = \{f_w : w \in [-W, W]^{l(d+2)}, \sigma(z) = 1/(1 + e^{-\gamma z}), 0 < W < \infty\}$$

where γ is called the *gain* parameter.

Sigmoid Feedfoward Networks - Sample Sizes

Performance Condition:

$$P\left[I(\hat{f}_w) - I(f_w^*) > \epsilon\right] < \delta.$$

Let $\mathcal{G}_{\mathcal{W}}^{\gamma} = \{(x - f_w(y))^2 : f_w \in \mathcal{F}_{\mathcal{W}}\}$ and $R = 8(A + lW)^2$.
Sample size I

$$\frac{16R}{\epsilon^2}\left(\ln(18/\delta) + 2\ln(8R/\epsilon^2) + \ln(2\gamma^2 W^2 lR/\epsilon) + \frac{\gamma W^2 lR}{\epsilon}\left[\left(\frac{\gamma W^2 lR}{\epsilon} - 1\right)^{N-1} + 1\right]\right)$$

Sample size II

$$\frac{16R}{\epsilon^2}\left(\ln(18/\delta) + 2\ln(8R/\epsilon^2) + l(N+2)\ln(L_w R/\epsilon)]\right)$$

where $L_w = \max(1, WB\gamma^2/4, W\gamma^2/4)$, or for $\gamma = 1$

Sample size III

$$\frac{128R}{\epsilon^2}\max\left\{\ln\left(\frac{8}{\delta}\right), [l(2N+3)+1]\ln\left(\frac{16e(l+1)R}{\epsilon}\right)\right\}$$

for $\epsilon > 0$, $0 < \delta < 1$.

Vector Space Methods

Function class $\mathcal{F}$ forms a finite-dimensional vector space

Motivation:

1. Sample complexity is simple function of the dimensionality

2. Computational problem is polynomial-time computable

3. Number of well-known examples belong to this class

Examples

1. **Potential Function Methods:** Aizerman *et al*
Basis of vector space $\{f_1, f_2, \ldots, f_d\}$
$f_i(y)$ is of the form $exp((y-\alpha)^2/\beta)$

2. **Special Feedforward Sigmoidal Networks:**
Proposed by Kurkova based on Kolmogorov's superposition theorem:
Network is of the form

$$\sum_{k=1}^{d} a_i \eta_i(y)$$

where $\eta_i(.)$'s are universal consists of sigmoidal networks.

3. **Polynomials:**
Fixed degree l polynomials form $(l+1)$-dimensional vector space

Dimensionality and Covering Number

Covering Number: $N(\epsilon, d_n, S)$: smallest number of closed balls of radius ϵ whose union covers S under pseudo metric d_n.

Previous Result: $\mathcal{F}$ is d-dimensional vector space
VC-dim of the sets of the form $\{\{x : f(x) \geq 0\} : f \in \mathcal{F}\}$ is upper bounded by d

Result of Haussler (1992):
For any probability measure P we have

$$N(\epsilon, d_P, \mathcal{F}) \leq 2\left(\frac{2e}{\epsilon}\ln\frac{2e}{\epsilon}\right)^d.$$

$$d_P(f_1, f_2) = \int_{y\in\Re^N} |f_1(y) - f_2(y)|dP$$

Sample Size

f^*: fusion rule minimizes expected error
$\hat{f}$: empirical best fusion rule
$\mathcal{F}$: vector space of dimension d and range $[0, 1]$.

Main Result:
Given an iid sample of size

$$\frac{512}{\epsilon^2}\left[d \ln\left(\frac{64e}{\epsilon} + \ln\frac{64e}{\epsilon}\right) + \ln(8/\delta)\right],$$

we have

$$P[I(\hat{f}) - I(f^*) > \epsilon] < \delta$$

Outline of Proof

Step 1: Vapnik (1982)

$$P\left\{I(\hat{f}) - I(f^*) > \epsilon\right\} \leq P\left\{\sup_{g\in\mathcal{G}} |P_n g - Pg| > \epsilon/2\right\},$$

where $\mathcal{G} = \{g(x,y) = (x - f(y))^2 : f \in \mathcal{F}\}$,
$Pg = \int g(x,y)dP$ and $P_n g = \frac{1}{n}\sum_{i=1}^{n} g(X_i, Y_i)$.

Step 2: For $g_1, g_2 \in \mathcal{G}$ such that $g_i(x,y) = (x - f_i(y))^2$, $i = 1, 2$, we have

$$|g_1(x,y) - g_2(x,y)| \leq |[2x - f_1(y) - f_2(y)][f_1(y) - f_2(y)]| \leq 2|f_1(y) - f_2(y)|$$

Step 3: Haussler (1992)

$$N(\epsilon, d_n, \mathcal{G}) \leq N(\epsilon/2, d_n, \mathcal{F}) \leq 2\left(\frac{4e}{\epsilon}\ln\frac{4e}{\epsilon}\right)^d.$$

Step 4: Pollard (1984); Lugosi and Zeger (1995)

$$P\left\{I(\hat{f}) - I(f^*) > \epsilon\right\} \leq 8\left(\frac{64e}{\epsilon}\ln\frac{64e}{\epsilon}\right)^d e^{-\epsilon^2 n/512}.$$

Step 4: Right hand side is upperbounded by δ for the given sample size n

Computational Problem

$\{f_1, f_2, \ldots, f_d\}$: basis of $\mathcal{F}$
$f \in \mathcal{F}$ can be written as $f(y) = \sum_{i=1}^{d} a_i f_i(y)$ for $a_i \in \Re$.

Empirical Cost:
Consider $\hat{f} = \sum_{i=1}^{d} \hat{a}_i f_i(y)$ such that $\hat{a} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_d)$ minimizes

$$I_{emp}(\vec{a}) = \frac{1}{n} \sum_{k=1}^{n} \left(X_k - \sum_{i=1}^{d} a_i f_i(Y_k) \right)^2,$$

where $\vec{a} = (a_1, a_2, \ldots, a_d)$.

Quadratic Programming Problem:
Now $I_{emp}(\vec{a})$ can be written in the following form:

$$I_{emp}(\vec{a}) = \frac{1}{n} \sum_{k=1} X_k^2 + \sum_{i=1}^{d} \sum_{j=1}^{d} a_i c_{ij} a_j + \sum_{i=1}^{d} a_i d_i$$

where

$$c_{ij} = \frac{1}{n} \sum_{k=1} f_i(Y_k) f_j(Y_k)$$

$$d_i = \frac{-2}{n} \sum_{k=1}^{n} f_i(Y_k) X_k.$$

Computational Problem

Quadratic Programming Problem:
$\hat{a} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_d)$ minimizes quadratic form

$$a^T C a + a^T d$$

$C = [c_{ij}]$ is a positive definite symmetric and $D = [d_i]$.

Result:
This problem is polynomial-time solvable using quadratic programming methods

Haar System

For $m = 0, 1, \ldots,$ Q_m:
diadic cubes $[0,1]^N = \bigcup_{J \in Q_m} J,\ J \cap J' = \emptyset$ for $J \neq J'$
$|J|$: N-dimensional volume of $J = 2^{-dN}$.

Define $P_m : L^\infty(Q) \mapsto L^\infty(Q)$ by

$$P_m f(x) = \frac{1}{|J|} \int_J f(y) dy$$

for $x \in J$ and $J \in Q_m$.

Critical Property:
The condition

$$\omega(f; r) = O(r^\alpha) \quad as \quad r \to 0_+$$

implies

$$\| f - P_m f \|_\infty = C/2^{\alpha m} \quad as \quad m \to \infty$$

for some $C > 0$.

Haar Kernel and Estimator

Haar kernel:
$P_m(x,y) = 2^{dm} \sum_{J \in Q_m} 1_J(x) 1_J(y)$ for $x, y \in Q$.

Density estimator (Ciesielski 1988):

$$\tilde{p}_{m,n} = \frac{1}{n} \sum_{j=1}^{n} P_m(x, X_j) = \sum_{J \in Q_m} n(J) h_J(x)$$

with $n(J) = \frac{1}{n} |\{j : X_j \in J\}|$ and $h_J(x) = \frac{1}{|J|} 1_J(x)$.

Nadaraya-Watson Estimator

Function estimator is

$$\hat{f}_{m,n}(y) = \frac{\sum_{j=1}^{n} X_j P_m(y, Y_j)}{\sum_{j=1}^{n} P_m(y, Y_j)} = \frac{\sum_{Y_j \in J} X_j}{\sum_{Y_j \in J} 1_J(Y_j)}$$

Note:

1. Nadaraya-Watson estimator specialized to Haar kernels.

2. Employed by Engel (1994) for fixed-design regression.

3. General Nadaraya-Watson estimator is well-known in statistics.

Nadaraya-Watson - Sample Size

Conditions on Functions:
1. Set of functions $\mathcal{F} \subseteq \mathcal{C}([0,1]^N)$ with range $[0,1]$
2. $\omega_\infty(f;r) \leq kr$ for some $0 < k < \infty$.

Conditions on Distributions:
There exists a family of densities $\mathcal{P} \subseteq \mathcal{C}([0,1]^d)$;
1. for each $p \in \mathcal{P}$, $\omega_\infty(p;r) \leq kr$ and
2. there exists $\mu > 0$ such that for each $p \in \mathcal{P}$, $p(x) > \mu$.

Sample Size:

$$\frac{2^{2m+4}}{\epsilon_1^2}\left[\left(\frac{k2^m}{\epsilon_1}\left[\left(\frac{k2^m}{\epsilon_1}-1\right)^{N-1}+1\right]+m\right)\right.$$
$$\left.\ln\left(2^{m+1}k/\epsilon_1\right)+\ln\left(\frac{2^{2m+6}}{(\delta-\lambda)\epsilon_1^4}\right)\right]$$

where $\epsilon_1 = \epsilon(\mu-\epsilon)/4$, $0 < \beta < \frac{N}{2(N+1)}$, $m = \lceil \frac{\log n\beta}{N} \rceil$ and $\lambda = b\left(\frac{2}{\epsilon}\right)^{1/N+1-1/2\beta} + b\left(\frac{2}{\epsilon_1}\right)^{1/N+1-1/2\beta}$.

Performance: $P\left[I(\hat{f}_{m,n}) - I(f^*)| > \epsilon\right] < \delta$

Computational Complexity

Note: $\hat{f}_{m,n}(y)$ can be computed in $O(n)$ time

Preprocessing:

Organize the data into N-range tree:
time complexity $O(n(\log n)^{N-1})$
for each J store sum and number of points;

Computation of estimator value at x:
sum and number of points in J containing x can be retrieved in $O((\log n)^N)$ time.

$\hat{f}_{m.n}(y)$ can be computed in $O((\log n)^N)$ time.

Note:
N-range tree is well-known in computational geometry and information retrieval.

Fusion of Noisy Function Estimators

System of 5 function estimators

$g(X)$, $X \in [0,1]^d$: Unknown function

Realized by a feedforward neural network

Estimator i: for $i = 1, 2, \ldots, 5$

$$g_i(X) = g(X)(1/2 + iZ/10)$$

Z: uniformly distributed over $[-1, +1]$

Fusion Function: $f : [0,1]^5 \mapsto [0,1]$

$f(g_1(X), \cdots, g_5(X))$ must approximate $g(X)$

Question:

Can a better predictor be obtained by "fusing" the noisy ones ?

Simulation Results
Fusion of Noisy Function Estimators

Mean square error:

Training Set	Testing Set	Nadaraya-Watson	Nearest Neighbor	Neural Network
100	10	0.000902	0.002430	0.048654
1000	100	0.001955	0.003538	0.049281
10000	1000	0.001948	0.003743	0.050942

(a) $d = 3$

Training Set	Testing Set	Nadaraya-Watson	Nearest Neighbor	Neural Network
100	10	0.004421	0.014400	0.018042
1000	100	0.002944	0.003737	0.021447
10000	1000	0.001949	0.003490	0.023953

(b) $d = 5$

Presentation Outline

1. Generic Sensor Fusion Problem

1.1 Formulation and Examples

1.2 Finite Sample Solutions

1.3 Performance Issues

1.4 Applications

2. Detection/Classification Problems

2.1 **Distributed Detection Problem**

2.2 **Sample-Based Solutions**

3. Is Fusion Easier ?

3.1 Isolation Fusers

3.2 Projective Fusers

4. Metafusers

4.1 Fusion of Fusers

4.2 Practical Paradigm

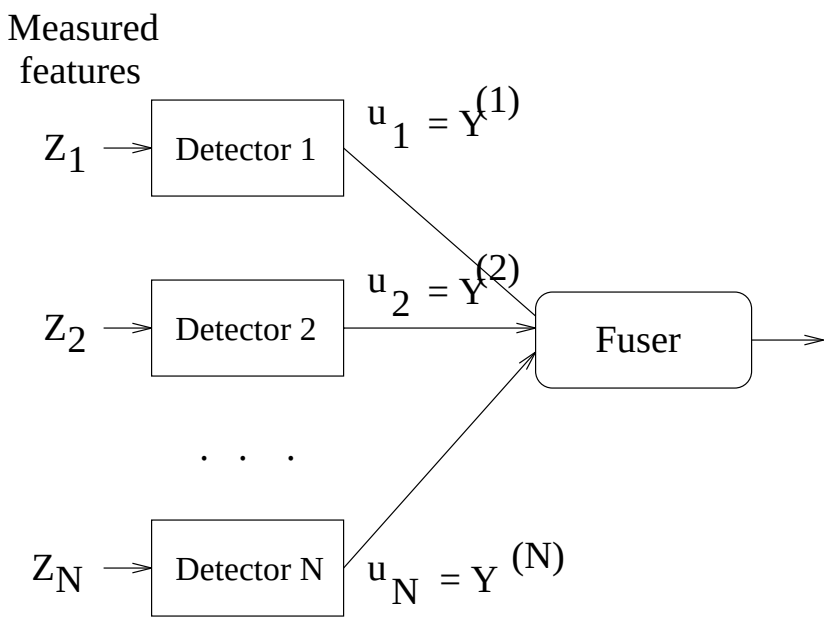
5. Performance Issues

5.1 Finite Sample Guarantees

5.2 Computational Issues

5.3 Practical Problems

Detection Problem: Parallel Sensor Suite



Detector System: $D_1, D_2, \ldots, D_N$

Detector: D_i
Makes a decision $u_i \in \{H_0, H_1\}$

Fusion Center:
Receives $u = (u_1, u_2, \ldots, u_N)$ outputs either H_0 or H_1

Notes:

1. Well-studied problem in different domains:
   democracy models (Condorcet 1786)
   composite methods (Laplace 1818)
   reliability (von Neumann 1956)
   pattern recongition (Chow 1965)
   Newer applications are being found in diverse areas
2. Of particular importance to distributed sensor systems
   Extensively studied over the past decade
   Dasarathy (1994), Varshney (1996), Rao et al (1996)

Examples

1. **Intruder Detection System:**
   Detectors monitor workspace from different vantage points
   Each detector is equipped with sensors, algorithms
   H_0: intruder is present; H_1: intruder is not present
   D_i: u_i is generated probabilistically; $P_i(u|H_0)$, $P_i(u|H_1)$
   Question: Can individual results be combined to obtain more reliable decision ?

2. **DNA Analysis System:** (Uberbacher and Mural 1991)
   Each detector is a software program that examines a segment of human DNA sequence
   H_0: Segment is protein-coding region; H_1: otherwise
   Question: Can different programs be combined to obtain improved performance ?

Note:
In the examples, systems are available
— data can be collected.

Example Fusion Rule: (Hashlamoun and Varshney, 1993)
Average-cost criterion is optimized by the likelihood ratio test

$$T(u) = \frac{P(u|H_1)}{P(u|H_0)} > \frac{\pi_0(C_{10} - C_{00})}{\pi_1(C_{01} - C_{11})} \tag{T.1}$$

where
C_{kj}: cost of deciding H_k when H_j is true, $k, j = 0, 1$

Motivation

In our formulation:

(i) Probability distributions are NOT known, and

(ii) Detector system is available, hence experimental samples can be collected.

Note:

1. Number of robotic applications belong to this formulation
2. Cost of obtaining probability distributions can be traded for some loss of performance

Existing results:

For many formulations of problem, fusion rules have been derived under the complete knowledge of the distributions

Our Results:

The fusion rules designed under known distributions can be implemented when only a sample is available.

Empirical Implementation

Fusion rules : expressed in terms of
(i) probabilities $p = (p_1, p_2, \ldots, p_n)$
(ii) data $u = (u_1, u_2, \ldots, u_N)$
of the form

$$R(p, u) > 0, \tag{T.2}$$

where

$$\text{decision is} \begin{cases} H_1 & \text{if the inequality is true} \\ H_0 & \text{otherwise.} \end{cases}$$

If probabilities are known:
$R(p, u)$ for given u can be explicitly evaluated.

In our formualtion:

1. Estimators $\hat{p}_i$ are computer based on iid sample
   $(u^1, H^1), (u^2, H^2), \ldots, (u^l, H^l)$
2. Empirical version given by

$$R(\hat{p}, u) > 0$$

is employed

Question: How good is $R(\hat{p}, u)$ compared to $R(p, u)$?

<u>Performance Criteria</u>

<u>Performance Measure:</u> Expected error of $R(\hat{p}, u)$,

$$E_u[|\Theta[R(p,u)] - \Theta[R(\hat{p},u)]|] = \sum_u |\Theta[R(p,u)] - \Theta[R(\hat{p},u)]|P(u),$$

where $\Theta[x]$ is 1 if x is non-negative and 0 otherwise.

<u>Performance Criterion:</u>
$R(\hat{p}, u)$ implements $R(p, u)$ with <u>confidence</u> $1 - \lambda$ if

$$P[\Theta[R(p,u)] \neq \Theta[R(\hat{p},u)]] < \lambda$$

or equivalently

$$E_u[|\Theta[R(p,u)] - \Theta[R(\hat{p},u)]|] < \lambda$$

for sufficiently (but finite) large sample of size $l < \infty$.

Informally,
based on a sufficiently large sample, $R(p, u)$ and $R(\hat{p}, u)$ yield the same result with a probability of at least $1 - \lambda$.

Independent Hypotheses

Formulation of Chair and Varshney (1986):
(i) independent detectors, (ii) a priori distributions are known
Fusion rule is of the form, for $n \geq 1$

$$\prod_{i=1}^{n} q_i - \prod_{i=1}^{n} s_i > 0 \tag{T.3.1}$$

where q_i and s_i are the probabilities of suitable events Q_i and S_i.

Empirical implementation of (T.3.1):

$$\prod_{i=1}^{n} \hat{q}_i - \prod_{i=1}^{n} \hat{s}_i > 0$$

where $\hat{q}_i$ and $\hat{s}_i$ are empirical estimates of q_i and s_i respectively based on the l-sample.

Theorem 3.1:
For any $r > 2$, consider a sample of size

$$l = \left\lceil \frac{1}{2\epsilon_L^2} \ln(2/\delta) \right\rceil$$

where $\epsilon_L = \left(1 + \frac{|\prod_{i=1}^{n} \hat{q}_i - \prod_{i=1}^{n} \hat{s}_i|}{r+2}\right)^{1/n} - 1$. Empirical implementation of $\prod_{i=1}^{n} q_i - \prod_{i=1}^{n} s_i > 0$ has confidence

$$1/2^{2n} - \delta(1 - 1/2^{2n})$$

or

$$1 - 2n\delta$$

Neyman-Pearson Test

Thomopoulos et al. (1989): a prori probabilities not known
Fusion rule is Neyman-Pearson test in the form

$$\prod_{i=1}^{n} q_i - \tau \prod_{i=1}^{n} s_i > 0 \tag{T.3.2}$$

where the positive real τ is fixed by the type I and II errors.

Corollary 3.1:
Consider a sample of size

$$l = \left\lceil \frac{1}{2\epsilon_L^2} \ln(2/\delta) \right\rceil$$

where $\epsilon_L = \left(1 + \frac{|\prod_{i=1}^{n} \hat{q}_i - \tau \prod_{i=1}^{n} \hat{s}_i|}{1+\tau}\right)^{1/n} - 1$. Empirical implementation of $\prod_{i=1}^{n} q_i - \tau \prod_{i=1}^{n} s_i > 0$ has confidence confidence

$$1 - 2n\delta$$

or

$$1/2^{2n} - \delta(1 - 1/2^{2n})$$

Lipschitz Test

$R(p, u)$ is Lipschitz with respect to p:
there exists a positive constant L such that

$$|R(p + \Delta p, u) - R(p, u)| < L||\Delta p||$$

for all Δp, u, where $||\Delta p||$ denotes the Euclidean norm of Δp in $\Re^n$.

Note:

1. $R(p, u)$ must be continuous in p
2. L is "upperbounded" by maximum gradient magnitude wrt p

Current Fusion Rules:

1. Large majority of published fusion rules are Lipschitz
2. There are several examples of non-Lipschitz tests
   some of these tests can be approximated by Lipschitz tests

Sample Size for Lipschitz Test

Theorem 3.2
Consider a decision rule $R(p, u)$ with Lipschitz constant L.
For any $r \geq 2$, given a training sample of size

$$l = \left\lceil \frac{r^2 n L^2}{2(R(\hat{p}, u))^2} \ln(2/\delta) \right\rceil$$

$R(\hat{p}, u) > 0$ implements the test $R(p, u) > 0$ with confidence

$$1 - n\delta$$

or

$$1/2^n - \delta(1 - 1/2^n)$$

Proof Outline: Theorem 3.2

1. Conditions $\sup_i |p_i - \hat{p}_i| < \epsilon$ and $|R(p)| \geq L\sqrt{n}\epsilon$, ensure that $R(p,u) > 0$ and $R(\hat{p},u) > 0$ yield the same result.

2. For $\epsilon = \frac{|R(p,u)|}{L\sqrt{n}}$, given a sample of size

$$l = \left\lceil \frac{1}{2\epsilon^2} \ln(2n/\delta) \right\rceil \tag{3.2.1}$$

   $R(\hat{p},u)$ implements $R(p,u)$ with the required precision.

3. Lower bound for ϵ by noting that $2|R(p,u)| \geq |R(\hat{p},u)|$ which implies $|R(p,u)| \geq \frac{1}{r}|R(\hat{p},u)|$ for any $r \geq 2$.

4. The sample size is obtained by using the lower bound $\frac{|R(\hat{p},u)|}{rL\sqrt{n}}$ for ϵ.

Note:
Tighter sample bounds may be possible in particular cases,
e.g. Theorem 3.1

Neyman-Pearson Test

Particular form of the test, for a positive real τ,

$$P(u|H_1) - \tau P(u|H_0) > 0 \qquad (T.3.4)$$

Let $\hat{P}(A)$ be fraction of times event A took place in the sample.

(i) Given a training sample of size

$$l = \left\lceil \frac{4(1+\tau)^2}{[\hat{P}(u|H_1) - \tau\hat{P}(u|H_0)]^2} \ln(4/\lambda) \right\rceil$$

the empirical rule $\hat{P}(u|H_1) - \tau\hat{P}(u|H_0) > 0$ implements the test $P(u|H_1) - \tau P(u|H_0) > 0$ with confidence $1-\lambda$.

(ii) Given a training sample of size

$$l = \left\lceil \frac{72(1+\tau)^2}{[\hat{P}(u \cap H_1)\hat{P}(H_0) - \tau\hat{P}(u \cap H_0)\hat{P}(H_1)]^2} \ln(8/\lambda) \right\rceil$$

the empirical test $\hat{P}(u \cap H_1)\hat{P}(H_0) - \tau\hat{P}(u \cap H_0)\hat{P}(H_1) > 0$ implements the test $P(u|H_1) - \tau P(u|H_0) > 0$ with confidence $1-\lambda$.

Correlation Coefficients Method:

Drakopoulos and Lee (1991):
Method of correlation coefficients

$$\mathcal{C} = \{P[u_{i_1}...u_{i_k}|H_j]|\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, N\}, j = 0, 1\}.$$

The fusion test is given by

$$P(u|H_1) - \tau P(u|H_0) > 0 \tag{T.3.5}$$

for suitable τ such that for $j = 0, 1$

$$P(u|H_j) = \sum_{I \subseteq A_0} (-1)^{|I|} P\left[\prod_{i \in A_1 \cup I} u_i|H_j\right]$$

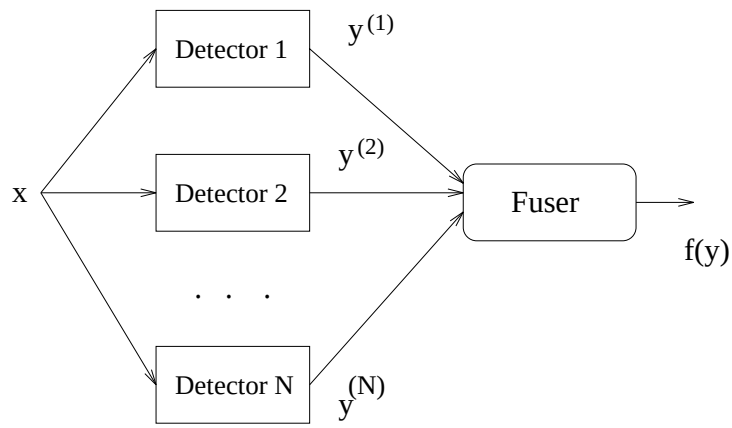
where $A_k = \{i : u_i = k\}$ and I, of cardinality $|I|$, varies over all subsets of A_0.

Corollary 3.3:
Given a training sample of size

$$l = \left\lceil \frac{r^2 2^{3N-1}(1+\tau)^2}{[\hat{P}(u|H_1) - \tau\hat{P}(u|H_0)]^2} \ln(2/\delta)) \right\rceil$$

the empirical rule implements $P(u|H_1) - \tau P(u|H_0) > \tau$ with confidence $1/2^N - \delta(1 - 1/2^N)$.

Simulation Results: Decision Fusion



Five detectors D_i, $i = 1, 2, \ldots 5$
Input: Boolean with equal probability

Detector D_i:
output is input with prob. $1 - i/10$;
opposite with prob. $i/10$;
Detectors are statistically independent.
$\pi_0 = \pi_1 = 1/2$

Simulation Results: Decision Fusion

Sensor	Probability of Correct Classification
S_1	90.0%
S_2	80.0%
S_3	70.0%
S_4	60.0%
S_5	50.0%

Note:

1. Best classifier:
   correct classification probability - 90%
2. Bayesian fuser:
   empirical performance - 91%

3. Both nearest neighbor and empirical fuser also achieve 91% correct classification.

Decision Fusion (Cntd.)

Percentage of correct classification:

Sample Size	Test set size	Bayesian Fuser	Empirical Decision	Nearest Neighbor	Nadaraya-Watson
100	100	91.91	23.00	82.83	88.00
1000	1000	91.99	82.58	90.39	89.40
10000	10000	91.11	90.15	90.81	91.42
50000	50000	91.19	90.99	91.13	91.14

Bayesian Fuser: Uses probability distribution
(Chair and Varshney 1986)

$$\left.\begin{array}{l}\text{Empirical Decision}\\ \text{Nearest Neighbor}\\ \text{Nadaraya} - \text{Watson}\end{array}\right\}\ \text{Use only the sample}$$

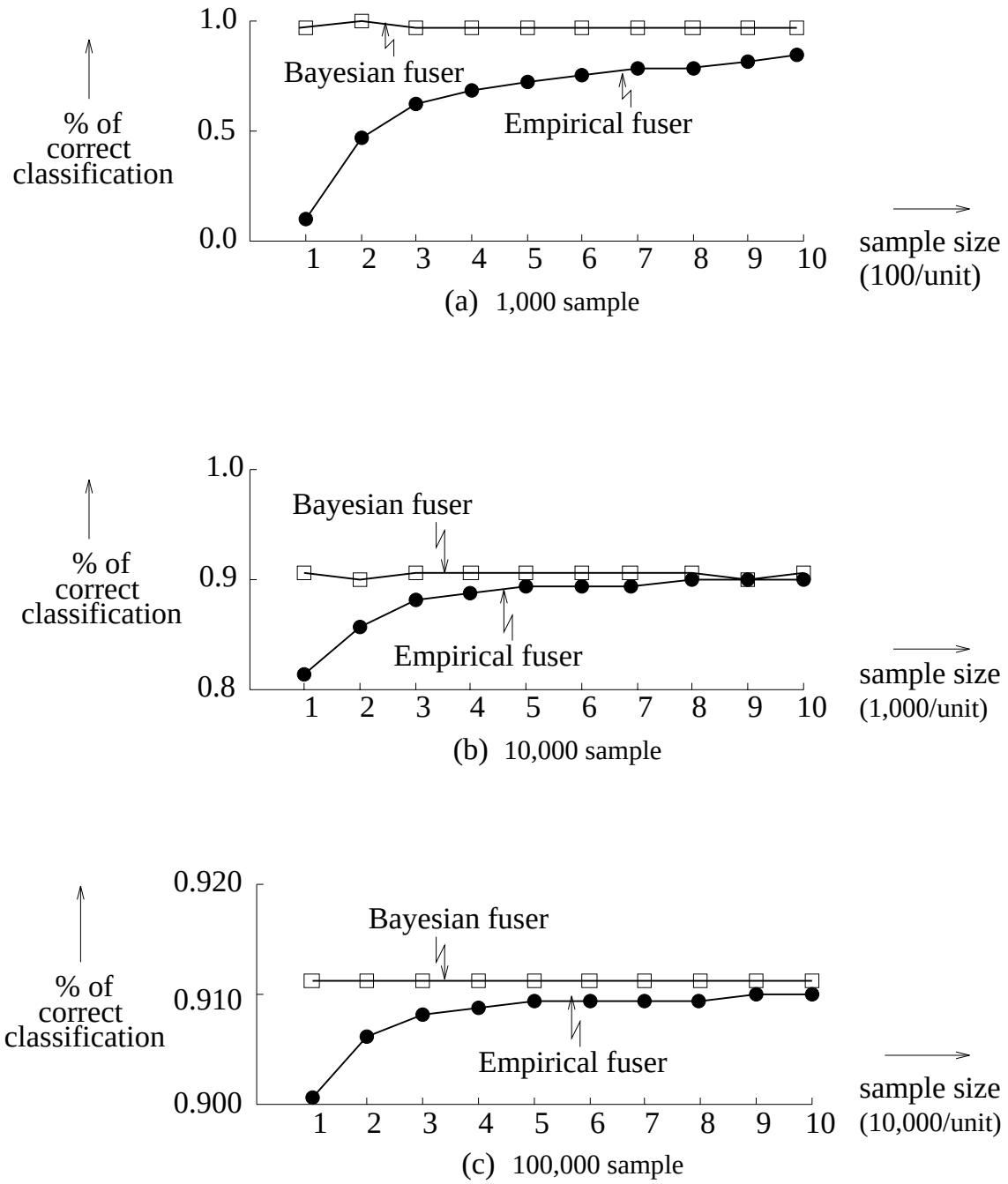


Figure 1: Relative performance of the Bayesian fuser and empirical fuser with training.

Decision Fusion (Cntd.)

Percentage of misclassification:

Sample Size	Test set size	S_1	S_2	S_3	S_4	S_5	Nadaraya-Watson
100	100	7.0	20.0	33.0	35.0	55.0	12.0
1000	1000	11.3	18.5	29.8	38.7	51.6	10.6
10000	10000	9.56	20.19	30.38	39.82	49.68	8.58
50000	50000	10.038	20.136	29.854	39.904	50.050	8.860

Note:
The fuser performs better than the best estimator S_1 after 1000 examples.

Presentation Outline

1. Generic Sensor Fusion Problem

1.1 Formulation and Examples

1.2 Finite Sample Solutions

1.3 Performance Issues

1.4 Applications

2. Detection/Classification Problems

2.1 Distributed Detection Problem

2.2 Sample-Based Solutions

3. Is Fusion Easier ?

3.1 **Isolation Fusers**

3.2 **Projective Fusers**

4. Metafusers

4.1 Fusion of Fusers

4.2 Practical Paradigm

5. Performance Issues

5.1 Finite Sample Guarantees

5.2 Computational Issues

5.3 Practical Problems

Multiple Sensor System

Sensor System: $S_1, S_2, \ldots, S_N$

Sensor: S_i
$Y^{(i)} \in [0,1]$: output of S_i for input $X \in [0,1]$:

generated according to probability distribution $P_{Y^{(i)}|X}$

Sensor Distributions: $P_i(.)$'s are unknown.

Fusion Rule:
Given sensor outputs $(Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)})$
Compute input X

Fusion Function: Special case: $f : [0,1]^N \mapsto [0,1]$

<u>Performance of Sensor</u>

Expected error of sensor S_i

$$I\ \ (S_i) = \int [X - Y^{(i)}]^2 dP_{Y^{(i)},X},$$

where $P_{Y^{(i)}|X}$ is the distribution of S_i.

<u>Best Sensor:</u>

$$S_{i^*} = \arg\min_{i=1}^{N} I\ \ (S_i)$$

<u>Note:</u>

1. If the distributions are known,
   best sensor can be obtained in principle

2. Here distributions are unknown, but a sample is known

Performance of Fuser

Expected error of fusion function f

$$I(f) = \int [X - f(Y)]^2 dP_{Y,X},$$

where $P_{Y|X}$ is the joint distribution of $S_1, \ldots, S_N$

Best Fuser:
From a family of functions $\mathcal{F}$

$$f^* = \arg\min_{f \in \mathcal{F}} I(f)$$

Example:
$\mathcal{F}$: Set of all sigmoid neural networks

Note:
1. If the distributions are known,
   best fuser can be obtained in principle
2. Here distributions are unknown, but a sample is known
3. Known distributions can be incorporated by adjusting the cost.

Fuser Versus Best Sensor

Question 1: If distributions are known,
are we better off choosing one of the sensors ?

$$\text{Answer}: \quad I\ (S_{i^*}) < I_F(f^*)?$$

Question 2: If only a sample is known,
are we better off by choosing one of the sensors ?

Short Answer:
If $\mathcal{F}$: (a) satisfies isolation property,
(b) "small" cover-size,
then, with high probability fuser is better.

Informally,
(a) fuser should be able to pick best sensor as a minimum, and
(b) must do it efficiently.

Isolation Property

$\mathcal{F} = \{f : [0,\tau]^k \mapsto [0,\tau]\}$ has the isolation property if it contains the functions

$$f^i(y_1, y_2, \ldots, y_k) = y_i$$

for all $i = 1, 2, \ldots, k$.

Important Property:

$$I_F(f^*) = \min_{i=1}^{N} I\ \ (S_i) - \Delta$$

for some $\Delta \in [0, \infty)$.

Note:
(1) precise value of Δ depends on $\mathcal{F}$,
(2) isolation property guarantees that

$$I_F(f^*) \leq \min_{i=1}^{N} I\ \ (S_i)$$

as a minimum.
Informally, f^* is at least as good as best sensor.

Examples:
1. Linear combinations have isolation property
2. Sigmoid feedforward networks do not have isolation property

Covering Numbers

Set S be equipped with a pseudometric ρ.

Covering Number: $N_C(\epsilon, \rho, S)$ under metric ρ:
smallest number of closed balls of radius ϵ, and centers in S, whose union covers S.

Function Classes: $\mathcal{G} = \{g : \Re^M \mapsto [0, 1]\}$
We consider two metrics: for $g_1, g_2 \in G$ we have

$$d_P(g_1, g_2) = \int_{z \in \Re^M} |g_1(z) - g_2(z)| dP,$$

for the probability distribution P defined on $\Re^M$, and

$$d_\infty(g_1, g_2) = \sup_{z \in \Re^M} |g_1(z) - g_2(z)|.$$

Note:
Alternate characterizations exist for approximation properties of $\mathcal{F}$.

<u>Main Result:</u>

If $\mathcal{F}$ has the isolation property, we have

$$I_F(f^*) = \min_{i=1}^{N} I\ \ (S_i) - \Delta,$$

for $\Delta \in [0, \infty)$, and

$$P_{Y,X}^{l}\left[I_F(\hat{f}) - \min_{i=1}^{N} I\ \ (S_i) + \Delta > \epsilon\right] < \delta$$

given the sample size l of at least

$$\frac{2048}{\epsilon^2}[\ln N_C(\epsilon/64, \mathcal{F}) + \ln(4/\delta)]$$

for cases:
(i) $N_C(\epsilon, \mathcal{F}) = N_C(\epsilon, d_\infty, \mathcal{F})$, and
(ii) $N_C(\epsilon, \mathcal{F}) = N_C(\epsilon, d_P, \mathcal{F})$ for all distributions P.

Linear Combination Fusers

- Linear combinations satisfy isolation property.

- One of the widely-used fuser methods, for fusing classifiers, regression estimators, and neural networks.

Simple Sample Size:

$$\frac{2048}{\epsilon^2}\left[N \ln\left(\frac{128e}{\epsilon}\ln\frac{128e}{\epsilon}\right) + \ln(8/\delta)\right]$$

Classification Problem

Special Case: Each S_i is a target detector or classifier:
$X \in \{0,1\}$, $Y^{(i)} \in \{0,1\}$, and

$$I\ (S_i) = \int [X \oplus Y^{(i)}] dP_{Y^{(i)},X}$$

corresponds to the probability of misclassification of S_i

Finite Cover Size: $|\mathcal{F}| \leq 2^{2^N}$,
since $\mathcal{F}$ at most consists of all Boolean functions on N variables
— $N_C(\epsilon, \mathcal{F}) \leq |\mathcal{F}|$ for all $\epsilon > 0$.

<u>Feedforward Networks</u>

Single hidden layer of h nodes and a single output node.
Output for input $y \in [0,1]^d$ is given by

$$f_w(y) = \sum_{j=1}^{h} a_j \sigma(b_j^T y + t_j)$$

where $w = (w_1, w_2, \ldots, w_{h(d+2)})$ is the <u>weight vector</u> consisting of
a_1, a_2, ..., a_h,
$b_{11}, b_{12}, \ldots, b_{1d}, \ldots,$ $b_{h1}, \ldots, b_{hd}$, and
$t_1, t_2, \ldots, t_h$, and $\sigma(z)$ is given by

$$\sigma^\tau(z) = \begin{cases} z & \text{if } z \in [0,\tau] \\ \tau & \text{if } z > \tau \\ 0 & \text{if } z < 0 \end{cases}$$

for some bounded $\tau > 1$.

Function Class:

$$\mathcal{FL}_B = \{f_w$$

<u>Potential Functions</u>

$f \in \mathcal{F}$ is of the form

$$w_1\phi_1(x) + \ldots + w_s\phi_s(x),$$

for $w_i \in \Re$, $x \in \Re^d$, $\phi_i : \Re^d \mapsto \Re$, corresponding to the basis $\{\phi_1, \ldots, \phi_s\}$.

Computation of Estimator:
$\hat{f}_i$ corresponds to the weight vector $(w_1, \ldots, w_s)$ that minimizes the empirical error.

Sample Size:
$\mathcal{F}_i$ forms a vector space of dimensionality s:

$$\frac{2048}{\epsilon^2}\left[s \ln\left(\frac{128e}{\epsilon} \ln \frac{128e}{\epsilon}\right) + \ln(8/\delta)\right]$$

<u>Note:</u>
1. general potential functions do not satisfy the isolation property,
2. piecewise linear version of ϕ_i's satisfy isolation property,
3. computation of $\hat{f}$ is polynomial-time

Asymptotic Results

In statistics literature,
asymptotic consistency results are more common.

Consistency:

The estimate $\hat{f}$ is consistent if

$$I_F(\hat{f}) \to I_F(f^*)$$

with probability one as $l \to \infty$.

Under finiteness of $N_C(\epsilon, \mathcal{F})$, consistency result follows from Borel-Cantelli Lemma if

$$\sum_{l=1}^{\infty} 4N_C(\epsilon/64, \mathcal{F})e^{-l\epsilon^2/2048} < \infty$$

for every $\epsilon > 0$.

This condition is true since

$$\sum_{l=1}^{\infty} e^{-l\epsilon^2/2048} \quad \leq \quad \frac{2048}{\epsilon^2} e^{-\epsilon^2/2048}$$

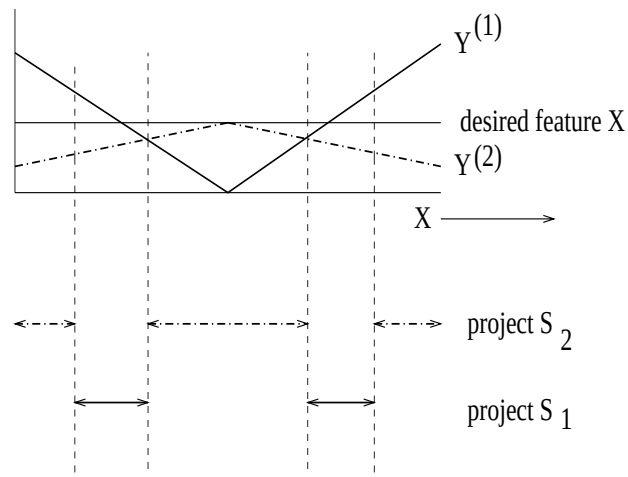
which is finite for all $\epsilon > 0$.

Class of Projective Fusers

Projective Fuser: f_P
— corresponds to a *partition* $P = \{\pi_1, \pi_2, \ldots, \pi_k\}$, $k \leq N$, of input space $\Re^d$
— assigns each block π_i to a sensor S_j such that

$$f_P(Y) = Y^{(j)}$$

for all $X \in \pi_i$ i.e. fuser simply transfers output of S_j for every point in π_i.



Optimal Projective Fuser: f_{P^*}:
— minimizes $I(.)$ over all partitions of $\Re^d$, and
— assignments of blocks to sensors $S_1, S_2, \ldots S_N$.

Projective Fuser Based on Error Regressions

Error Regression of S_i and f_F:

$$\mathcal{E}(X, S_i) = \int C(X, Y^{(i)}) dP_{Y|X}$$

$$\mathcal{E}(X, f_P) = \int C(X, f_P(Y)) dP_{Y|X}$$

Projective Fuser Using Error Regressions:

$$f_{LE}(Y) = Y^{(i_{LE}(X))}$$

where

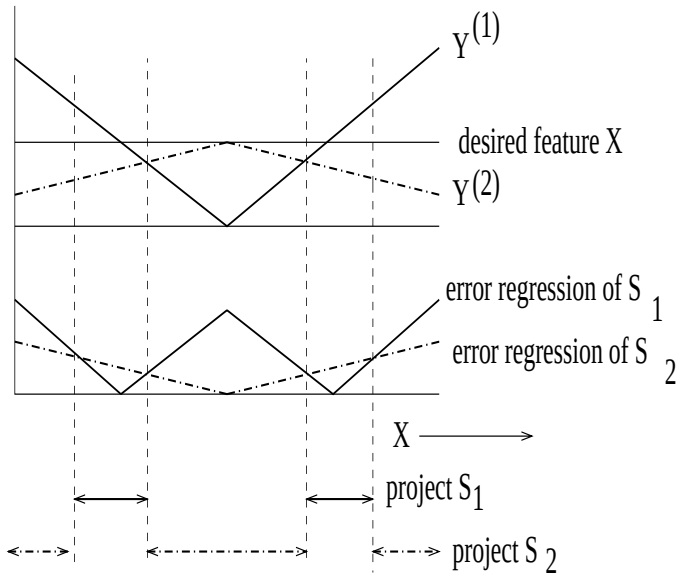
$$i_{LE}(X) = \arg \min_{i=1,2,\ldots,N} \mathcal{E}(X, S_i)$$

Projective Fuser Based on Error Regressions

Property:

Error regression of f_{LE} is lower envelope of the set of error regressions of sensors given by $\{\mathcal{E}(X, S_1), \ldots, \mathcal{E}(X, S_N)\}$.

$$\mathcal{E}(X, f_{LE}) = \min_{i=1,\ldots,N} \mathcal{E}(X, S_i)$$



Example 1

X : uniformly distributed over $[0,1]$ – measured by two sensors S_1 and S_2:

$$Y^{(1)} = X + |X - 1/2| + U$$

$$Y^{(2)} = X + 1/[4(1 + |X - 1/2|)] + U$$

where U is an independent random variable with zero mean

For $C(X, Y^{(i)}) = (X - Y^{(i)})^2$, error regressions are given by

$$\mathcal{E}(X, S_1) = (X - 1/2)^2 + E[U^2]$$

$$\mathcal{E}(X, S_2) = 1/[16(1 + |X - 1/2|)^2] + E[U^2]$$

Comparison between sensors:

$$I(S_1) = 0.0833 + E[U^2] \quad \text{and} \quad I(S_2) = 0.125 + E[U^2],$$

– S_1 is the better of the two sensors.

Example 1 (Cntd.)

Projective Fuser f_{LE}:
Corresponds to lower envelope of $\mathcal{E}(X, S_1)$ and $\mathcal{E}(X, S_2)$.

range for X	sensor to be projected
$[0, 0.134]$	S_2
$[0.134, 0.866]$	S_1
$[0.866, 1]$	S_2

$$I(f_{LE}) = 0.0828 + E[U^2]$$

Classification Example

$X \in [0,1] \times \{0,1\}$ is specified by a function $f_X = 1_{[1/4,3/4]}$
X is generated as follows: $X = (Z, f_X(Z))$

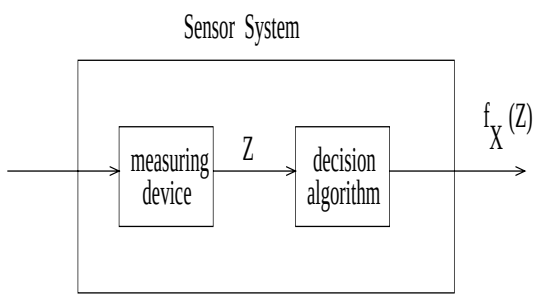
1. Z is uniform in $[0,1]$ — first component
2. $f_X(Z)$ forms the second component

Detection Problem:

Z: Use sensor to measure a feature
f_X: Function used by classification algorithm
Second component of X:
— presence ($f_X(Z) = 1$) or absence ($f_X(Z) = 0$) of target
— represented by a feature Z taking a value in the interval $[1/4, 3/4]$

Sensor System:

1. device to measure the first component Z of X
2. algorithm to compute the second component.



Classification Example - Cntd.

Sensor system:
S_1 and S_2 have:

— ideal devices to measure Z without an error

— make errors in utilizing the measured features

$$Y^{(1)} = (Z, 1_{[1/4-\epsilon_1, 3/4]}(Z))$$

$$Y^{(2)} = (Z, 1_{[1/4, 3/4-\epsilon_2]}(Z))$$

for some $0 < \epsilon_1, \epsilon_2 < 1/4$

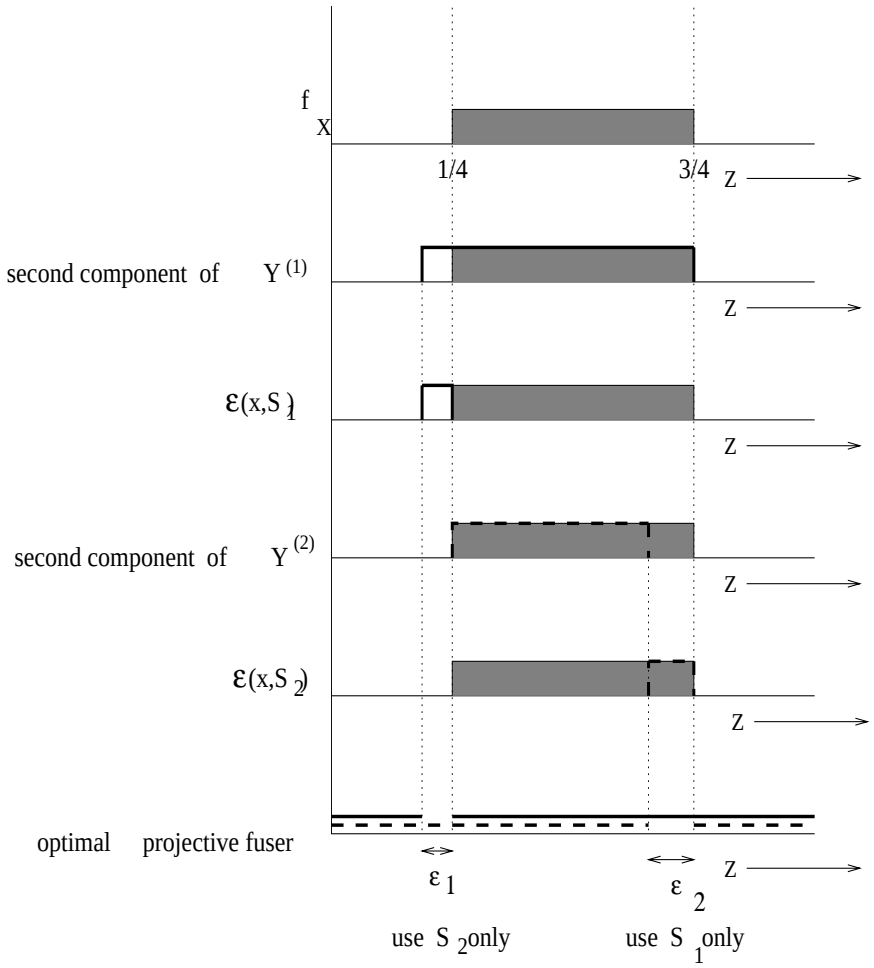
For quadratic cost function $C(X, Y^{(i)}) = (X - Y^{(i)})^T (X - Y^{(i)})$, error regressions:

$$\mathcal{E}(X, S_1) = 1_{[1/4-\epsilon_1, 1/4]}(Z)$$

$$\mathcal{E}(X, S_2) = 1_{[3/4-\epsilon_2, 3/4]}(Z)$$

$I(f_{LE}) = 0$, where as both $I(S_1)$ and $I(S_2)$ are positive.

Classification Example - Cntd.



Optimality Result

Theorem:
The projective fuser based on the lower envelope of error regressions is optimal among all projective fusers.

Proof: Y: sensor output corresponding to X
For any projective fuser $f_P(Y)$:
$i_P(X)$: index of the estimator such that $f_P(Y) = Y^{(i_P(X))}$
For $X \in \Re^d$, we have

$$\begin{aligned} E[C(X, f_P(X))|X] &= \mathcal{E}(X, f_P) \\ &= \mathcal{E}(X, Y^{(i_P(X))}) \\ &\geq \mathcal{E}(X, f_{LE}) \\ &= E[C(X - f_{LE}(X))|X]. \end{aligned}$$

By taking the expectations with respect to P_X on both sides, we have

$$I(f_{LE}) \leq I(f_P)$$

<u>Linear Fusers</u>

For sensor output $Y = (Y^{(1)}, \ldots, Y^{(N)})$, linear fuser is defined as

$$f_L(Y) = \sum_{i=1}^{N} \alpha_i Y^{(i)}$$

for some $(\alpha_1, \ldots, \alpha_N) \in \Re^N$.

Optimal Linear Combination Fuser: f_{L^*}:
— minimizes $I(.)$ over all linear combinations

<u>Comparison with:</u> f_{LE}
1. If individual sensors perform well in localized regions,
— f_{LE} is better
2. If the sensors are equally distributed around certain values in global sense,
— f_{L^*} performs better

Better Projective Fuser

For classification example, for $f_L = \alpha_1 Y^{(1)} + \alpha_2 Y^{(N)}$, we have

$$\begin{aligned} I(f_L) &= \alpha_1^2 \int\limits_{[1/4-\epsilon_1,1/4)} dP_Z \\ &+ (1-\alpha_1-\alpha_2)^2 \int\limits_{[1/4,3/4-\epsilon_2)} dP_Z \\ &+ (1-\alpha_1)^2 \int\limits_{[3/4-\epsilon_2,3/4]} dP_Z \end{aligned}$$

which is non-zero no matter what the coefficient are.

Projective Fuser:

Error regressions of S_1 and S_2 are non-zero in $[1/4-\epsilon_1, 1/4]$ and $[3/4-\epsilon_2, 3/4]$ — intervals are disjoint:

error of one sensor being canceled by a scaler multiplier of the other.

In General:

– If the error regressions of sensors take non-zero values on disjoint intervals, then any linear fuser will have non-zero error.

– Disjointness yields $\mathcal{E}(X, f_{LE}) = 0$, for all X, and hence $I(f_{LE}) = 0$.

<u>Better Linear Fuser</u>

In classification example, let $f_X = 1$ for $Z \in [0,1]$,

$$Y^{(1)}(X) = (Z, \epsilon Z + 1 - \epsilon)$$

$$Y^{(2)}(X) = (Z, -\epsilon Z + 1 + \epsilon),$$

for $0 < \epsilon < 1$.

<u>Optimal linear fuser:</u>

$$f_{L^*}(Y) = 1/2(Y^{(1)} + Y^{(2)}) = 1$$

$$I(f_{L^*}) = 0$$

<u>Optimal projective fuser:</u> At every $X \in [0,1]$, we have

$$\mathcal{E}(X, S_1) = \mathcal{E}(X, S_2) = \epsilon^2 (1 - Z)^2 = \mathcal{E}(X, f_{LE}).$$

Thus, $I(f_{LE}) = \epsilon^2 \int_{[0,1]} (1 - Z)^2 dP_Z > 0$, whereas $I(f_{L^*}) = 0$.

Sample-Based Projective Fusers

Consider that sensor distributions are not available
— utilize regression estimator $\hat{\mathcal{E}}(X, S_i)$ of $\mathcal{E}(X, S_i)$
— lower envelope of the estimators in the computation of fuser.

Estimator:
For a sequence $\{h_l\}$ of positive numbers, consider the partition of $\Re$ given by

$$\theta_l = \{[(r-1)h_l, rh_l) | r \in \mathcal{Z}\}.$$

Let $\psi_l[X]$ denote the unique cell of θ_l that contains X. Then, the estimator of $\mathcal{E}(X, S_i)$ is given by

$$\hat{\mathcal{E}}(X, S_i) = \frac{\sum_{j=1}^{l} C(X_j, Y_j^{(i)}) 1_{\psi_l[X]}(X_j)}{\sum_{j=1}^{n} 1_{\psi_l[X]}(X_j)}.$$

Consider the conditions:

(i) $C(X, Y) < K$ for some $K > 0$;
(ii) $\lim_{l \to \infty} h_l \to 0$; and
(iii) $nh_l \to \infty$ as $l \to \infty$.

Known Result:
$\int |\mathcal{E}(X, S_i) - \hat{\mathcal{E}}(X, S_i)|^2 dP_X \to 0$ with probability 1 regardless of the distributions of sensors.

Consistency of Projective Fusers

Sample-Based Fuser:
The fuser $\hat{f}_{LE}$ is as previously defined with $\hat{\mathcal{E}}(X, S_i)$ used in place of $\mathcal{E}(X, S_i)$.

Consistency:
$I(\hat{f}_{LE}) \to I(f_{LE})$ as $l \to \infty$ with probability 1 for any sensor distributions.

Consistency Proof:
Since $\hat{f}_{LE}$ is a projective fuser, we have $I(\hat{f}_{LE}) \geq I(f_{LE})$. By definition $\hat{\mathcal{E}}(X, \hat{f}_{LE}) \leq \hat{\mathcal{E}}(X, f_{LE})$. Then,

$$\begin{aligned}
& I(\hat{f}_{LE}) - I(f_{LE}) \\
& \leq \int [\mathcal{E}(X, \hat{f}_{LE}) - \mathcal{E}(X, f_{LE})] dP_X \\
& \leq \int [\mathcal{E}(X, \hat{f}_{LE}) - \hat{\mathcal{E}}(X, \hat{f}_{LE})] dP_X + \int [\hat{\mathcal{E}}(X, \hat{f}_{LE}) - \mathcal{E}(X, f_{LE})] dP_X \\
& \leq \int |\mathcal{E}(X, \hat{f}_{LE}) - \hat{\mathcal{E}}(X, \hat{f}_{LE})| dP_X + \int |\hat{\mathcal{E}}(X, f_{LE}) - \mathcal{E}(X, f_{LE})| dP_X \\
& \leq 2 \sum_{i=1}^{N} \int |\mathcal{E}(X, S_i) - \hat{\mathcal{E}}(X, S_i)| dP_X.
\end{aligned}$$

Since $\int |\mathcal{E}(X, S_i) - \hat{\mathcal{E}}(X, S_i)| dP_X \to 0$, we have $I(\hat{f}_{LE}) \to I(f_{LE})$ as $l \to \infty$ with probability 1.

Non-Optimality in Larger Fuser Class

Negative Result:
f_{LE} may not be optimal in a larger class of fusers where some function of the sensor output can be projected.

Classification Example:
Consider $f_X = 1_{[1/4,3/4]}$,

$$Y^{(1)}(X) = (Z, 1_{[1/4-\epsilon_1,3/4-\epsilon_1]}(Z))$$

$$Y^{(2)}(X) = (Z, 1_{[1/4,3/4-\epsilon_2]}(Z))$$

for some $0 < \epsilon_1, \epsilon_2 < 1/8$, and $\epsilon_1 < \epsilon_2$. Thus, we have

$$\mathcal{E}(X, S_1) = 1_{[1/4-\epsilon_1,1/4]}(Z)$$

$$\mathcal{E}(X, S_2) = 1_{[3/4-\epsilon_2,3/4]}(Z)$$

whose lower envelope is not the zero function.

We have

$$\mathcal{E}(X, f_{LE}) = 1_{[3/4-\epsilon_2,3/4-\epsilon_1]}(Z)$$

$$I(f_{LE}) = \int_{[3/4-\epsilon_2,3/4-\epsilon_1]} dP_Z$$

By changing the assignment $Y^{(1)}$ of f_{LE} to $1-Y^{(1)}$ for $Z \in [3/4-\epsilon_2, 3/4-\epsilon_1]$, we achieve zero error.

Projective Fusers and Isolation Property

Projective fusers satisfy isolation property
— since g_i corresponds to entire $\Re^d$ forming one block assigned to the single sensor S_i.

Implication of Isolation Property:
$\mathcal{G} = \{g(Y)\}$ has isolation property,
we have for all $i = 1, 2, \ldots, k$,

$$\inf_{g \in \mathcal{G}} \int \mathcal{E}(X, g(Y)) dP_X \leq \int \mathcal{E}(X, Y^{(i)}) dP_X$$

which implies

$$\inf_{g \in \mathcal{G}} I(g) \leq \min_{i=1,\ldots,N} I(S_i)$$

Optimal Projective Fuser As Good As Best Sensor:
Since $I(f_{LE}) \leq \inf_{g \in \mathcal{G}} I(g)$
— f_{LE} performs at least as well as the best sensor.

Is Fusion Easier ?

Yes:
If fuser is just required to peform as at least as good as the best sensor.

No:
If finite sample guarantees are needed on the cost
– situation is very similar to classification:
Any fuser can perform poorly on bad distributions

Presentation Outline

1. Generic Sensor Fusion Problem

1.1 Formulation and Examples

1.2 Finite Sample Solutions

1.3 Performance Issues

1.4 Applications

2. Detection/Classification Problems

2.1 Distributed Detection Problem

2.2 Sample-Based Solutions

3. Is Fusion Easier ?

3.1 Isolation Fusers

3.2 Projective Fusers

4. Metafusers

4.1 **Fusion of Fusers**

4.2 **Practical Paradigm**

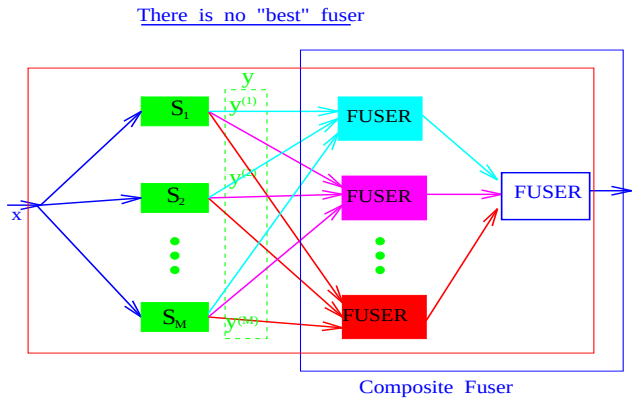
5. Performance Issues

5.1 Finite Sample Guarantees

5.2 Computational Issues

5.3 Practical Problems

Metafusers



Motivation:

1. Pick "best estimator" or fuse all/some estimators?

— old formulation

2. Can different fusers be fused ? — Yes, metafusers

— new formulation

— there is no single "best fuser"

Result: Informally,

if metafuser class $\mathcal{G}$ has isolation property

composite system is at least as efficient as the best elemental fuser or estimator

Metafuser 1

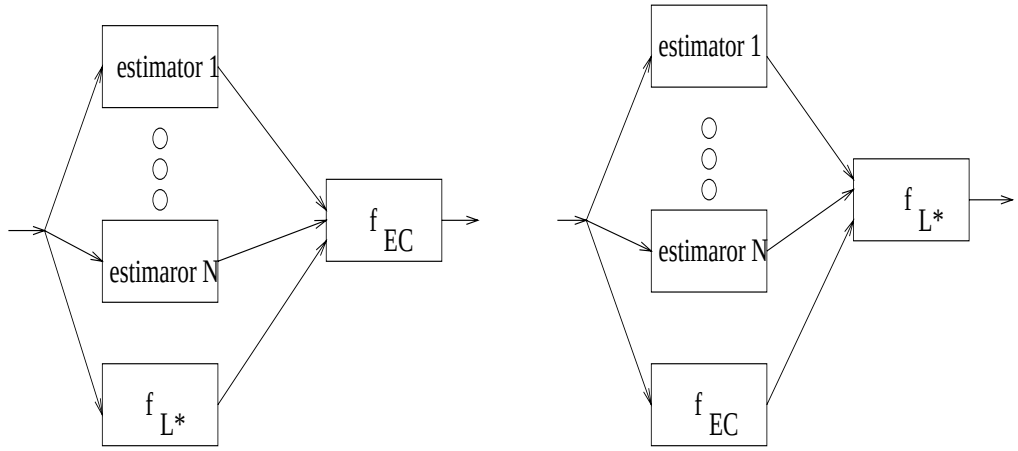
Include the optimal linear combination as S_{N+1}, we can guarantee that

$$I(f_{LE}) \leq I(f_{L^*})$$

by the isolation property of projective fusers

— linear combinations also satisfy the isolation property, we in turn have

$$I(f_{L^*}) \leq \min_{i=1,\ldots,N} I(S_i)$$



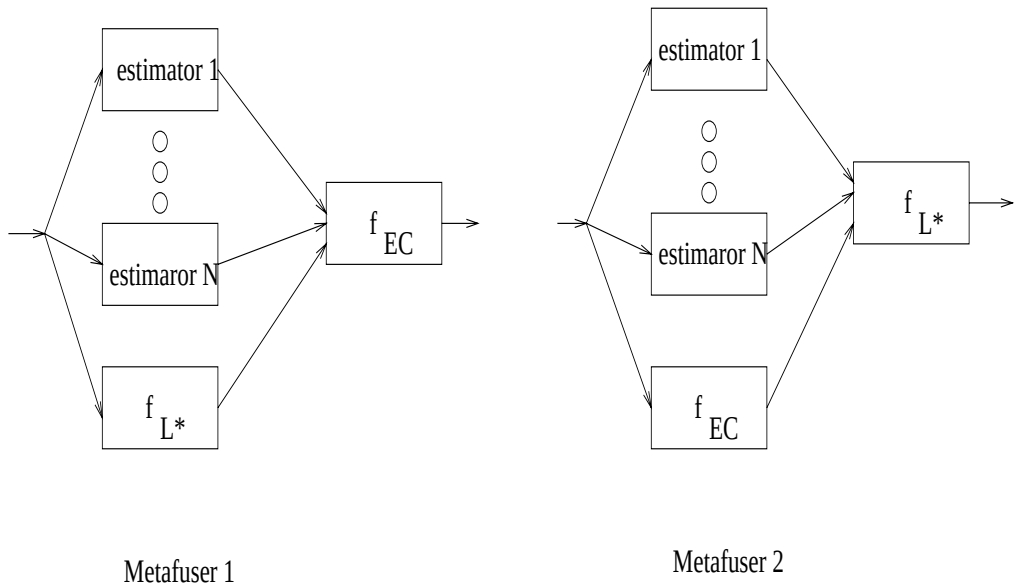
Metafuser 1

Metafuser 2

Metafuser 2

The roles of f_{L^*} and f_{LE} can be switched – by including f_{LE} as one of the components of f_{L^*} – to show that

$$I(f_{L^*}) \leq I(f_{LE}) \leq \min_{i=1,\ldots,N} I(S_i)$$

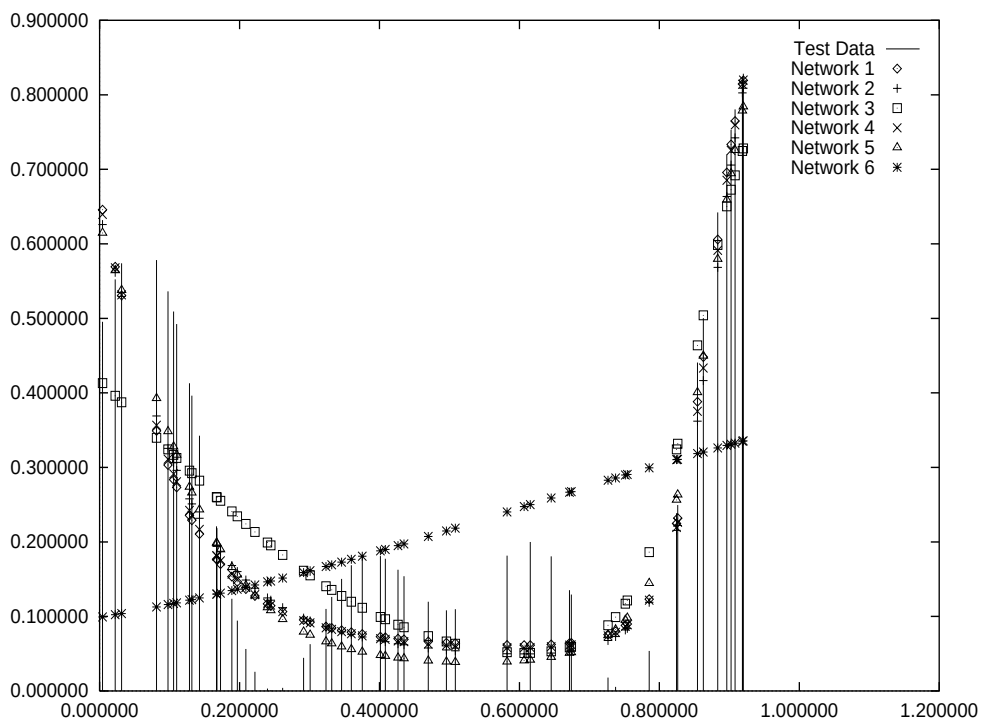


Simulation Results For Metafuser 1: Typical Results

Six neural network estimators for the target function

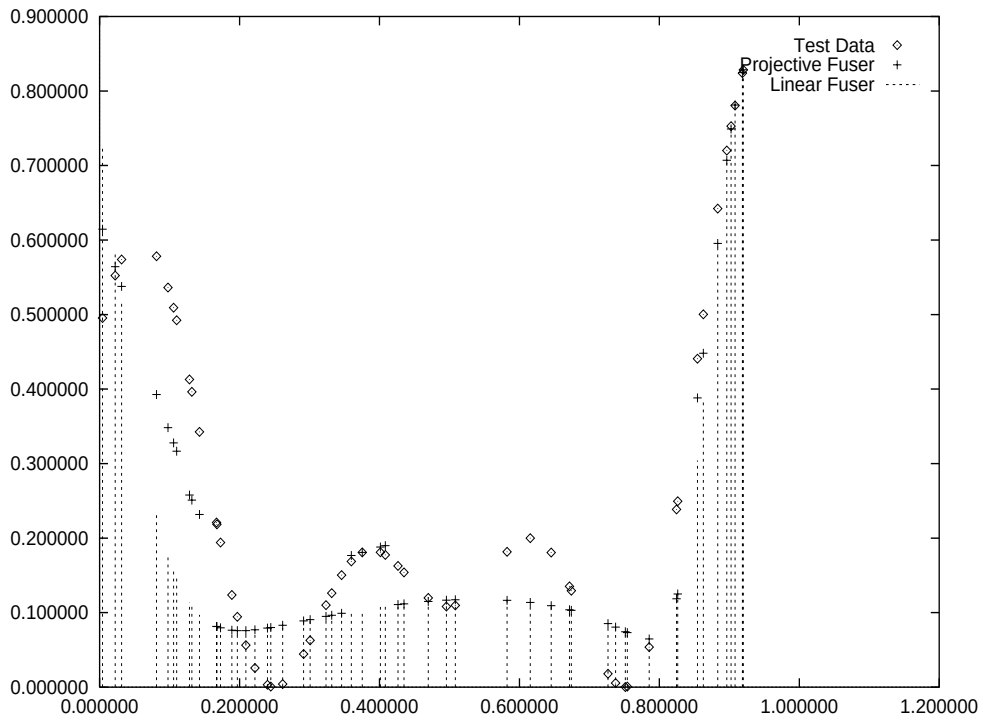
$$f_2(x) = 0.02(12 + 3x - 3.5x^2 + 7.2x^3)$$
$$(1 + \cos 4\pi x)(1 + 0.08 \sin 3\pi x).$$

Parameters:
– number of hidden nodes is randomly chosen
– trained with backpropagation algorithm with a different learning rate randomly chosen



Simulation Results For Metafuser 1: Fuser Results

Results: In terms of test error:
– Linear fusers is 31.15 times better than best estimator
– Projective fuser is 1.3 times better than linear fuser



Simulation Results For Metafuser 1: Summary

For each dataset size, 10 different samples are utilized

Results: In terms of test error:
Projective fuser outperformed both linear fuser and best estimator

data size		projective	other better		performance (times)		average
training	test	as good	linear	best	linear	best network	error
Without noise							
10	10	8	1	1	1.009269	10.489711	0.075042
25	25	8	2	0	1.039855	13.426878	0.021926
50	50	10	0	0	1.304039	31.157175	0.013454
75	75	10	0	0	1.530556	89.050201	0.004725
100	100	10	0	0	1.788104	87.905518	0.003764
With noise							
10	10	8	2	0	0.982823	9.205843	0.041874
25	25	8	2	0	1.045973	14.115362	0.026983
50	50	10	0	0	1.293410	19.121033	0.010399
75	75	9	1	0	1.275850	33.192585	0.008435
100	100	10	0	0	1.227069	37.937778	0.007115

A Practical Metafuser Paradigm

Include:
(i) Isolation fuser as one of the fusers
(ii) Metafuser with isolation property - linear combinations

New Sensor:
– Insert the sensor
– Retrain the isolation fuser and metafuser

Newfuser:
– Insert the fuser
– Retrain the metafuser

Result:
System performs at least as well as the best sensor and fuser at all times

Presentation Outline

1. Generic Sensor Fusion Problem

1.1 Formulation and Examples

1.2 Finite Sample Solutions

1.3 Performance Issues

1.4 Applications

2. Detection/Classification Problems

2.1 Distributed Detection Problem

2.2 Sample-Based Solutions

3. Is Fusion Easier ?

3.1 Isolation Fusers

3.2 Projective Fusers

4. Metafusers

4.1 Fusion of Fusers

4.2 Practical Paradigm

5. Performance Issues

5.1 **Finite Sample Guarantees**

5.2 **Practical Issues**

Finite Sample Guarantees

In most practical engineering systems:

Distributions:

(i) Either not known, or
(ii) Very expensive to estimate - expertise in multiple areas needed

Sample: Typically small
— Easier to obtain by experimentation
— More work is needed for providing practical guarantees
— Non-iid cases must to be considered

Computation: Computation of fuser must be efficient

Finite sample guarantees is a neccessity

A Perspective

Information Fusion arises in two flavors:

A. Fusion is part of Problem Specification:
System contains a number of information sources
— we need to combine the information

B. Fusion is Part of Solution:
Use multiple disparate solutions and combine

Fusion is Easier: if needs to be at least as good as best sensor
– Classifier needs to fix a class for finite sample guarantees
  even in theory we can do no better than "best-in-class"
– Fuser can be chosen from a class with isolation property

Diverse Sensors and Fusers: must be developed
– not sufficient to target a single method

Future Issues

Combination of Fusion and Tracking Methods:

— Both areas have seen significant advances
They must be combined.

Domain Specific Knowledge:

– must be exploited as much as possible

Newer Fusion Methods:

– must continue to be developed
– more diverse the better

Metafusers:

– Are isolation metafusers sufficient ?
– Trade-offs in fusion layers

Evaluation of Fusion Methods

Empirical Evaluation Methods:

– Systematic methods are needed: Theory and practice

Benchmarks:

– must be developed for each application class

Closing

Exciting Times for Sensor Fusion:

— several analytical advances
— novel and challenging applications
— very encouraging pratical results

It is only the beginning:

– Challenges in building practical working systems
– Developing sound and illuminating theories