Fusion of Continuous Output Classifiers

Jacob Hays
Amit Pillay
James DeFelice

Definitions

- x – feature vector
- c – number of classes
- L – number of classifiers
- $\{\omega_1, \omega_2, ..., \omega_c\}$ – Set of class labels
- $\{D_1, D_2, ..., D_L\}$ – Set of classifiers
  - All c outputs from D_i are in interval [0,1]
- DP(x) – Decision Profile matrix

$$DP(x) = \begin{bmatrix} d_{1,1}(x) & \dots & d_{1,j}(x) & \dots & d_{1,c}(x) \\ d_{i,1}(x) & \dots & d_{i,j}(x) & \dots & d_{i,c}(x) \\ d_{L,1}(x) & \dots & d_{L,j}(x) & \dots & d_{L,C}(x) \end{bmatrix}.$$

Approaches

- **Class Conscience -**Use one column of DP(x) at a time
  - Ex) Simple/Weighted Averages
- **Class Indifferent** – Treat DP(x) as a whole new feature space, Use new classifier to make final decision.

$$DP(x) = \begin{bmatrix} d_{1,1}(x) & \dots & d_{1,j}(x) & \dots & d_{1,c}(x) \\ d_{i,1}(x) & \dots & d_{i,j}(x) & \dots & d_{i,c}(x) \\ d_{L,1}(x) & \dots & d_{L,j}(x) & \dots & d_{L,C}(x) \end{bmatrix}.$$

Discriminant to Continuous

- Non-continuous classifiers produce label
- $\{g_1(x), g_2(x), \ldots g_c(x)\}$ – output of D
  - Would like to normalize to [0,1] interval
- $\{g'_1(x), g'_2(x), \ldots g'_c(x)\}$, where $\sum_{j=1}^{c} g'_j(x) = 1$
- **Softmax** Method Normalizes to [0,1] $g'_j(x) = \frac{\exp\{g_j(x)\}}{\sum_{k=1}^{c} \exp\{g_k(x)\}}$
- Better if g'(x) would be a probability

Converting Linear Discriminant

- Assuming normal densities

$$g_j(x) = \log\{P(\omega_j)\, p(x \mid \omega_j)\}$$

- Let C be the constant additive terms we drop

$$A = \exp\{C\} \qquad P(\omega_j)\, p(x \mid \omega_j) = A \exp\{g_j(x)\}$$

- Plug into Bayes Rule, and it simplifies to the softmax function

$$P(\omega_j \mid x) = \frac{A \times \exp\{g_j(x)\}}{\sum_{k=1}^{c} A \times \exp\{g_k(x)\}} = \frac{\exp\{g_j(x)\}}{\sum_{k=1}^{c} \exp\{g_k(x)\}}$$

Neural Networks

- Consider a NN, with c outputs, $\{y_1, \ldots, y_c\}$
- When trained using squared error rate, the outputs can be used for an approximation of posterior probability.
- Normalize them to [0,1] interval using softmax function.
- Normalization function independent of Neural network training, only occurs on outputs.

Laplace Estimator for Decision Tree

- In Decision Trees, you use entropy to split the distribution based on a single feature per level
- Normally, you continue to split until there is a single class in each leaf of the tree
- In **Probability Estimating Trees** , instead of splitting until a single class is in a leaf, split until around K points are in each leaf, and use various methods to calculate the probability of each class at each leaf.

Count based probability, Laplace

- $\{k_1, k_2, \ldots, k_c\}$ – Number of sample points of class $\{w_1, w_2, \ldots, w_c\}$ respectively in leaf
- $K = k_1 + k_2 + \ldots + k_c$
- Maximum Likelihood (ML) estimate of

$$\hat{P}(\omega_j \mid x) = \frac{k_j}{K}, \; j = 1, \ldots, c$$

- When K is too small, estimates are unpredictable

Laplace Estimator

- Laplace Correction

$$\hat{P}(\omega_j \mid x) = \frac{k_j + 1}{K + c}$$

- m-estimation:

$$\hat{P}(\omega_j \mid x) = \frac{k_j + m \times \hat{P}(w_j)}{K + m} -$$

- best to set m so

$$m \times \hat{P}(\omega_j) \approx 10$$

Ting and William Laplace estimator

- Ting and William
  - ω^* is majority class

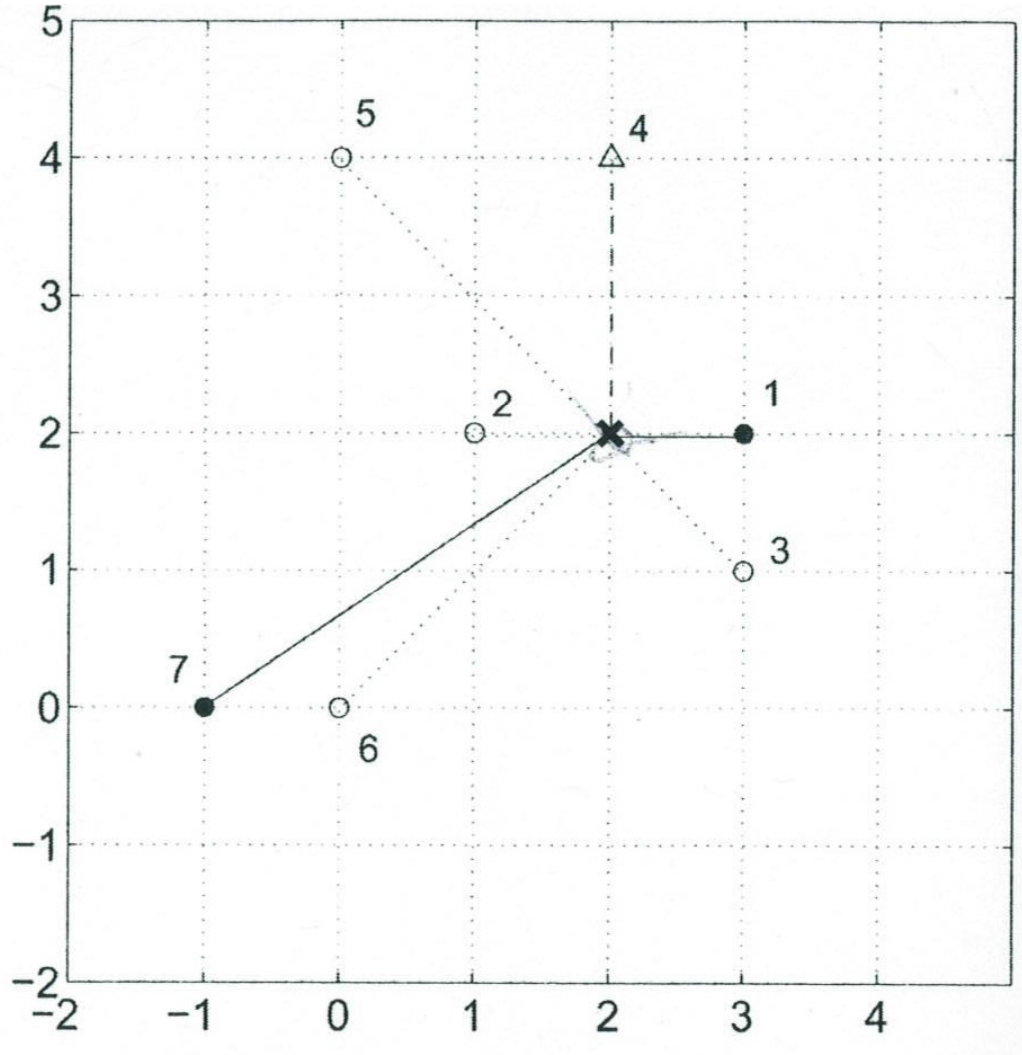
$$\hat{P}(\omega_j \mid x) = \begin{cases} 1 - \dfrac{\sum_{l \neq j} k_l + 1}{K + 2} & if\ (w_j = w^*) \\ \left[1 - \hat{P}(\omega^*)\right] \times \dfrac{k_j}{\sum_{l \neq j} k_l} & otherwise \end{cases}$$

Weighted Distance Laplace Estimate

- Take the average distance from x to all samples of class w_j, over the average distance to all samples

$$\hat{P}(\omega_j \mid x) = \frac{\sum_{x^{(j)} \in w_j} \frac{1}{d(x, x^{(j)})}}{\sum_{i=1}^{k} \frac{1}{d(x, x^{(i)})}}$$

Example



Method	$\mu_1(\mathbf{x})$	$\mu_2(\mathbf{x})$	$\mu_3(\mathbf{x})$
ML	$\frac{4}{7}$	$\frac{2}{7}$	$\frac{1}{7}$
Standard Laplace	$\frac{5}{10}$	$\frac{3}{10}$	$\frac{2}{10}$
m-estimation ($m = 12$)	$\frac{8}{19}$	$\frac{6}{19}$	$\frac{5}{19}$
Ting and Witten	$\frac{12}{27}$	$\frac{10}{27}$	$\frac{5}{27}$
Distance-based	0.58	0.30	0.12

Class Conscious Combiners

- Non-trainable Combiners
  - No extra parameters, all defined up front
  - Function of classifier output for specific class

$$\mu_j(x) = F[d_{1,j}(x), d_{2,j}(x), \ldots d_{L,j}(x)]$$

- Simple mean

$$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(x)$$

Popular Class Conscious Combiners

- Minimum/Maximum/Median

$$\mu_j(x) = \max_i \{d_{i,j}(x)\}$$

- Trimmed Mean:
  - L degrees of support sorted, X percent of values are dropped. Mean taken of remaining.
- Product

$$\mu_j(x) = \prod_{i=1}^{L} d_{i,j}(x)$$

Generalized Mean Function

$$\mu_j(x,\alpha) = \left(\frac{1}{L} \sum_{i=1}^{L} d_{i,j}(x)^{\alpha} \right)^{1/\alpha}$$

- Generalized Mean is defined as above except for the following special cases.
  - $a \rightarrow -\infty$, Minimum,
  - $a = -1$, Harmonic Mean
  - $a = 0$, Geometric mean
  - $a = 1$, Simple Arithmetic Mean
  - $a \rightarrow \infty$, Maximum

$$\mu_j(x) = \left(\prod_{i=1}^{L} d_{i,j}(x) \right)^{1/L}$$

- a is chosen before hand, level of optimism

Class Conscious Combiner Example

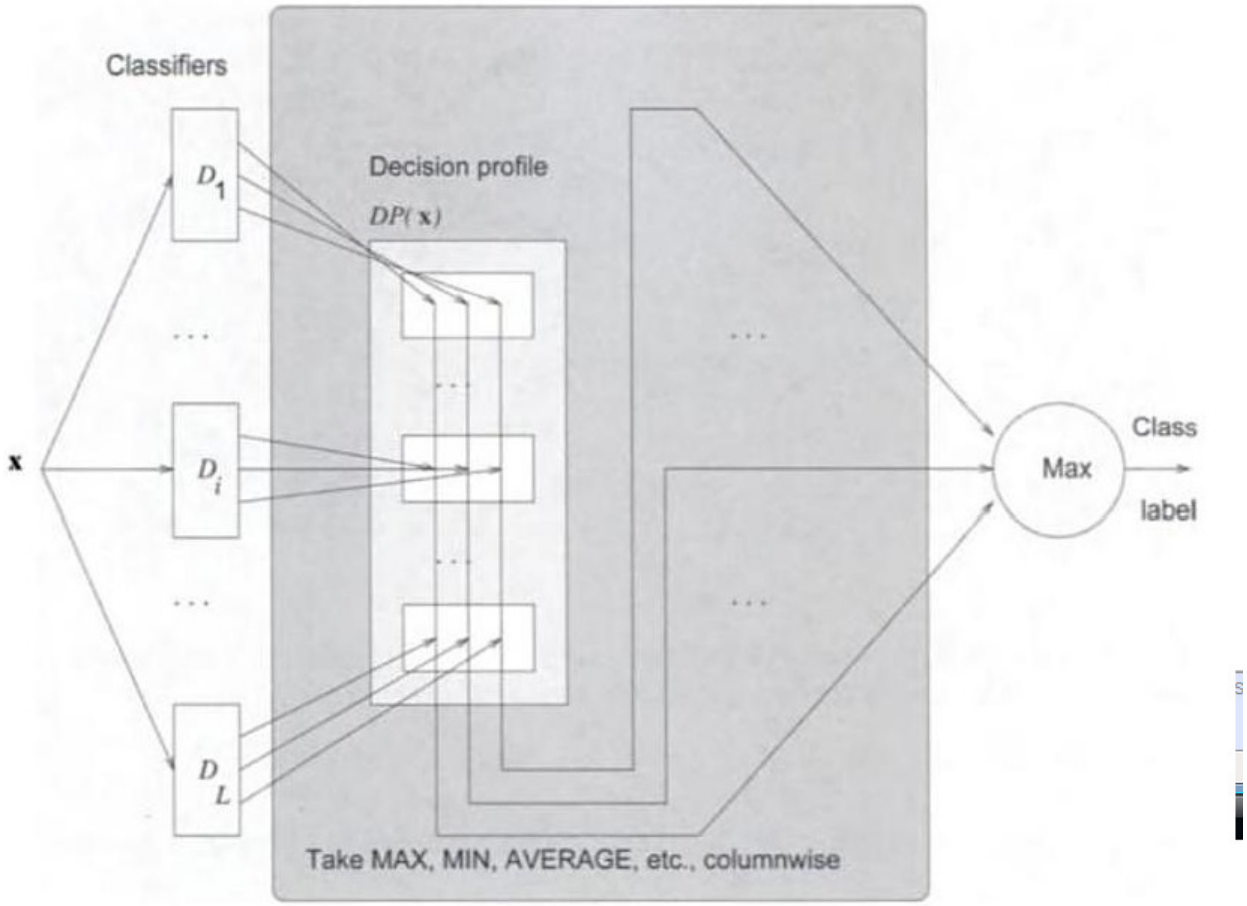
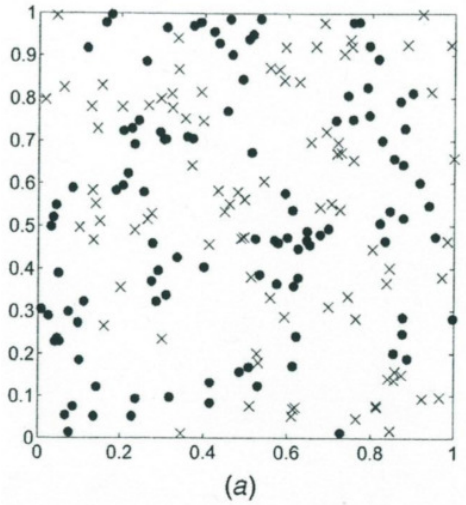
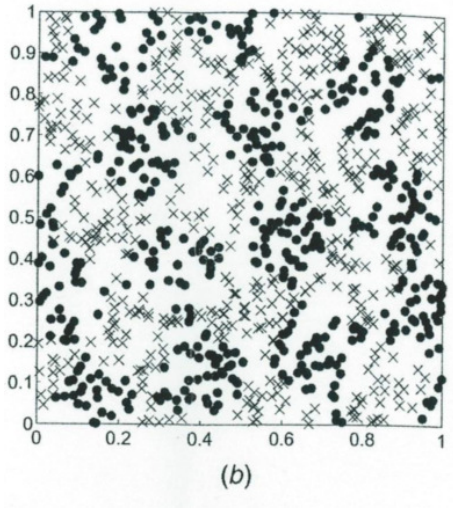


Fig. 5.2 *Operation of simple nontrainable (class-conscious) combiners.*

Example: Effect of Optimism α

- 100 training / test sets
  - Training set (a), 200 samples
  - Testing set (b), 1000 samples
- For each ensemble
  - 10 bootstrap samples (200 values)
  - Train classifier on each (Parzen)





Example: Effect of Optimism α

- Generalized mean
  - 50 <= α <= +50, steps of 1
  - -1 <= α <= +1, steps of 0.1
- Simple mean combiner gives best result

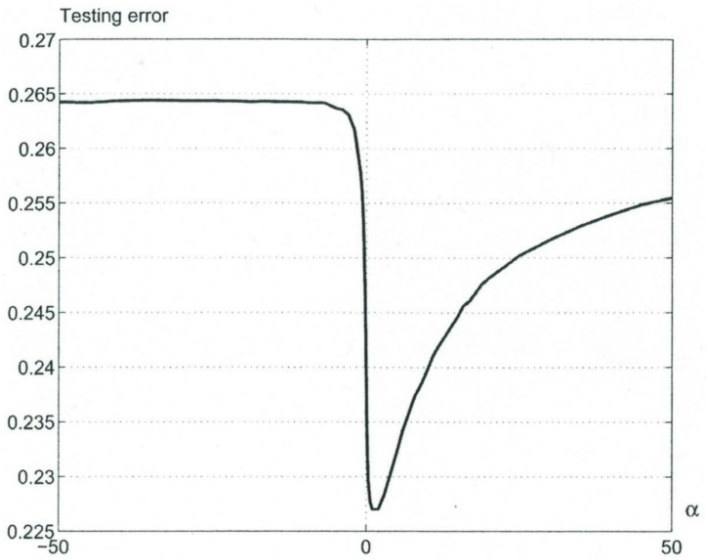


Fig. 5.4 *Ensemble error for the generalized mean combiner for $\alpha \in [-50, 50]$.*

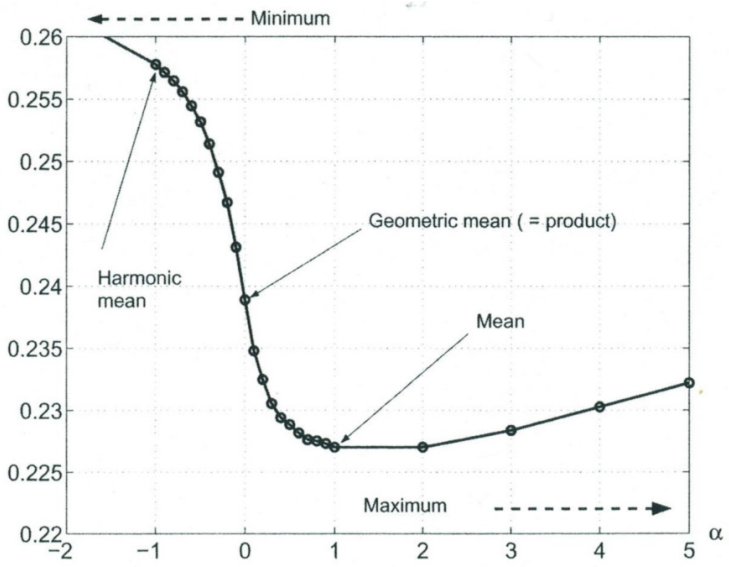


Fig. 5.5 *A zoom on the ensemble error for the generalized mean combiner for $\alpha \in [-2, 5]$.*

Interpreting Results

- Mean classifier isn't always the best
- Shape of the error curve depends upon
  - Problem
  - Base classifier used
- Average and product are most intensely studied combiners
  - For some problems, average may be…
    - Less accurate, but
    - More stable

Ordered Weight Averaging

- Generalized, non-trainable
- L coefficients (one for each classifier)
- Order the results of ω_j classifiers, descending
- Multiply by vector of coefficients b (weights)
  - $i_1, \ldots, i_L$ is a permutation of the indices 1, …, L

$$\mu(x) = \sum_{k=1}^{L} b_k d_{i[k]j}(x)$$

Ordered Weight Averaging: Example

- Consider a jury assessing sport performance (diving)
  - Reduce subjective bias
    - Trimmed mean
      - Drop lowest, highest scores
      - Average the remaining

$$d_j = [.6 \quad .7 \quad .2 \quad .6 \quad .6]^T$$

$$b = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

$$\mu_j = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix} [.7 \quad .6 \quad .6 \quad .6 \quad .2]^T = 0.6$$

Ordered Weight Averaging

- General form of trimmed mean
  - $b = [\ 0, 1/(L-2), 1/(L-2), \ldots, 1/(L-2), 0]^T$
- Other operations may be modeled with careful selection of b
  - Minimum: $b = [0, 0, \ldots, 1]^T$
  - Maximum: $b = [1, 0, \ldots, 0]^T$
  - Average: $b = [\ 1/L, 1/L, \ldots, 1/L]^T$
- Many resources spent on developing new aggregation connectives
  - Bigger question: when to use which one?

Trainable Combiners

- Combiners with additional parameters to be trained
  - Weighted Average
  - Fuzzy Integral

Weighted Average

- 3 groups, based on number of weights
- L-weights
  - One weight per classifier

$$\mu_j(x) = \sum_{i=1}^{L} w_i d_{i,j}(x)$$

  - Similar to equation we saw for ordered weight average, except we're trying to optimize w_i here (and we're not reordering $d_{i,j}$)
  - w_i for classifier D_i usually based on its estimated error rate

Weighted Average

- $c_x L$ weights
  - Weights are specific to each class

$$\mu_j(x) = \sum_{i=1}^{L} w_{ij} d_{i,j}(x)$$

  - Only j^{th} column is used in calc
  - Linear regression commonly used to derive optimal weights
  - “class conscious” combiner

Weighted Average

- $c_x c_x L$ weights
  - Support for each class determined from entire decision profile DP(x)

$$\mu_j(x) = \sum_{i=1}^{L} \sum_{k=1}^{c} w_{ikj} d_{i,k}(x)$$

  - Different weight space for each class ω_j
  - Whole decision profile is intermediate feature space
    - "class indifferent" combiner

Weighted Average: Class Conscious

$$\mu_j(x) = \sum_{i=1}^{L} w_{ij} d_{i,j}(x)$$

- $d_{i,j}(x)$ are point estimates of $P(\omega_j \mid x)$
  - If estimates are unbiased,
    - $\mu(x)$ is nonbiased minimum variance estimate of $P(\omega_j \mid x)$, conditional upon…
    - restriction of coefficients w_i to sum to 1

$$\sum_{i=1}^{L} w_i = 1$$

Weighted Average: Class Conscious

$$\mu_j(x) = \sum_{i=1}^{L} w_{ij} d_{i,j}(x)$$

- Weights derived to minimize variance of μ(x)
  - μ(x) variance <= variance of any single classifier
- We assume point estimates are unbiased
  - Variance of $d_{i,j}(x)$ = expected squared error of $d_{i,j}(x)$
- When coefficients w_i minimize variance
  - μ(x) is better estimate of $P(\omega_j \mid x)$ than any $d_{i,j}(x)$

Ex: Variance of Estimate of $P(\omega_j \mid x)$

- Calculate variance of $d_{i,j}(x)$ (estimates)
- Target values of $P(\omega_j \mid x)$ are
  - 1 (in class ω_j), 0 (not in class ω_j)

TABLE 5.1 Hypothetical Output for ω_1 from Two Classifiers for a Data Set $Z = \{z_1, \ldots, z_{10}\}$.

Data Point	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}
$d_{1,1}(z_k)$	0.71	0.76	0.15	0.09	0.15	0.62	0.98	0.56	0.44	0.79
$d_{2,1}(z_k)$	0.41	0.27	0.91	0.15	0.64	0.90	0.68	0.95	0.22	0.14
Target (ω_1)	1	1	1	0	0	0	0	0	0	0

  - Output for class w_1 of 2 classifier ensemble $D = \{D_1, D_2\}$, dataset $Z = \{z_1, z_2, \ldots, z_{10}\}$
  - First 3 points in w_1
  - Table shows first columns of 10 DPs

Ex: Variance of Estimate of $P(\omega_j \mid x)$

- Variance of D_i here is variance of approximation error
  - Approximation error determined as
    - $\{(1 - 0.71),(1 - 0.76),(1 - 0.15),\ldots,(0 - 0.79)\}$
  - Mean of approx. error for D_1 is -0.225
  - Variance of approx. error for D_1 is

$$\sigma^2 = \frac{1}{(10-1)}\left[(1-0.71+.225)^2 + \ldots + (0-0.79+.225)^2\right] \approx 0.32$$

  - Covariance matrix of approx. errors for classifiers is

$$\Sigma = \begin{bmatrix} 0.32 & 0.22 \\ 0.22 & 0.34 \end{bmatrix}$$

Constrained Regression

- Assume approximation errors are normally distributed, zero mean
  - $P(\omega_j \mid x) - d_{i,j}(x)$ = approximation errors
  - σ^2_{ik} is covariance of approximation errors between classifiers D_i, D_k
- General Legrange form $L(x,\lambda) = f(x) + \lambda g(x)$
- Find our optimal weights by minimizing J

$$J = \sum_{i=1}^{L}\sum_{k=1}^{L} w_i w_k \sigma_{ik} - \lambda\left(\sum_{i=1}^{L} w_i - 1\right)$$

Constrained Regression

$$J = \sum_{i=1}^{L}\sum_{k=1}^{L} w_i w_k \sigma_{ik} - \lambda\left(\sum_{i=1}^{L} w_i - 1\right)$$

- Solution for minimizing J
- where $w = \Sigma^{-1} I (I^T \Sigma^{-1} I)^{-1}$

$$w = [w_1, w_2, \ldots, w_L]^T$$

is our set of weights
I is vector of size L, all 1's

Ex: Constrained Regression

- Going back to the numbers we had from Table 5.1

$$\Sigma = \begin{bmatrix} 0.32 & 0.22 \\ 0.22 & 0.34 \end{bmatrix} \qquad w = \Sigma^{-1} I (I^T \Sigma^{-1} I)^{-1}$$

$$w = \begin{bmatrix} 5.6 & -3.6 \\ -3.6 & 5.3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 5.6 & -3.6 \\ -3.6 & 5.3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 0.54 \\ 0.46 \end{bmatrix}$$

  - All the weights and covariances need to be labeled with *j* to indicate which $P(\omega_j \mid x)$ we're estimating

Constrained Regression / Comparison

- Comparison of two combiners
  - Simple avg
  - Weighted avg
- L varied: 2 – 30
  - L > 20 tends to over fit

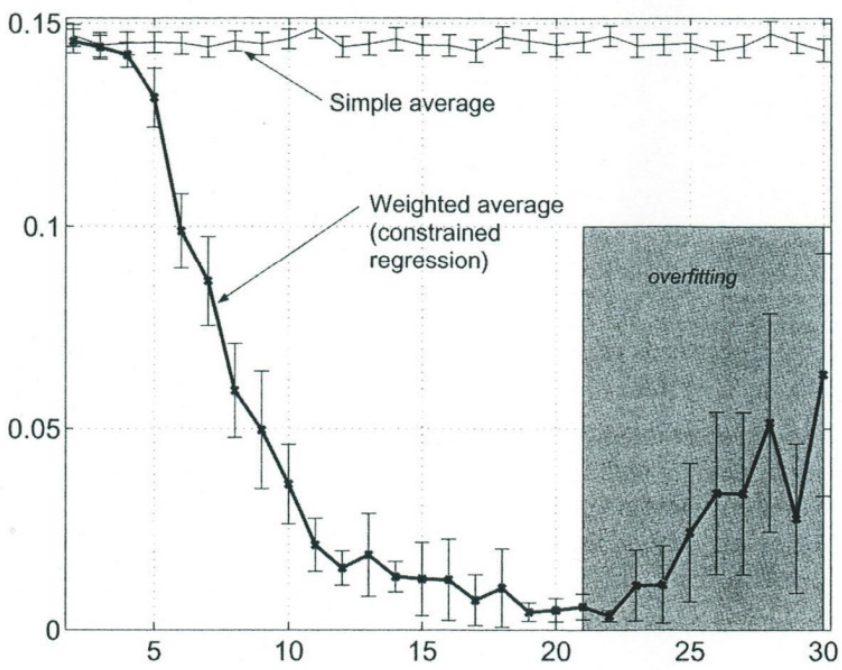


Fig. 5.6 *Weighted average (constrained regressions) and simple average on the banana data. The error bars mark the 95 percent confidence interval for the classification error calculated on the basis of 100 runs.*

Constrained Regression, Extension

- Suppose classifier outputs for ω_j are independent
  - Σ is diagonal, with variances for D_1, ... D_L along the diagonal
  - Simplifies weight optimization

$$w_i = \frac{\frac{1}{\sigma_i^2}}{\sum_{k=1}^{L} \frac{1}{\sigma_k^2}}$$

Fuzzy Integral

- Based on fuzzy set theory
- **Main Idea:**
  Measure the strength of not only for each classifier but also for all subset of classifiers
- Measure of strength of each subset of classifier gives how good this subset for the given input **x.** Also called fuzzy measure

Fuzzy Integral cont.

subset	D1	D2	D3	D1, D2	D1, D3	D2, D3	D1,D2, D3
g	0.3	0.1	0.4	0.4	0.5	0.8	1

*j*th column of decision profile for input **x** as [0.1 0.7 0.5]

Goal: To find $\mu_j(x)$

1. Sort the degrees of support in ascending order
2. Append 0 and 1 in the list if not present
3. For different value of α in the list find classifiers having support more than or equal to α
4. Subset of such classifiers are called α-cut (H_α)

Fuzzy Integral cont.

$\alpha = 0$	$H_0 = \{D1,D2,D3\}$	$g(H_0) = 1$
$\alpha = 0.1$	$H_{0.1} = \{D1,D2,D3\}$	$g(H_{0.1}) = 1$
$\alpha = 0.5$	$H_{0.5} = \{D2,D3\}$	$g(H_{0.5}) = 0.8$
$\alpha = 0.7$	$H_{0.7} = \{D2\}$	$g(H_{0.7}) = 0.1$
$\alpha = 1$	$H_1 = \Phi$	$g(H_1) = 0$

$$\mu_j(x) = max\{min(\alpha, g(H_\alpha))\} = max\{min(0,1), min(0.1,1), min(0.5,0.8), min(0.7,0.1), min(1,0)\}$$
$$= max(0, 0.1, 0.5, 0.1, 0) = 0.5$$

Class-Indifferent Combiners

- Unlike class-conscious combiners, these type of combiners uses all *Lxc* degrees of support in Decision Profile *DP(x)*

Decision Templates

- Typical decision profile (DP) that is a representative of class ω_j is called Decision Template (DT_j)

- Main Idea:
  - Compare decision template (DT_j) with the current decision profile DP(x) for some test input x using some similarity measure.

- Training:
  - For *j = 1 to c,* calculate the mean of the DPs(z_k) for inputs from some data set Z that belongs to class ω_j. This mean represent the decision template (DT_j)

Decision Template Cont.

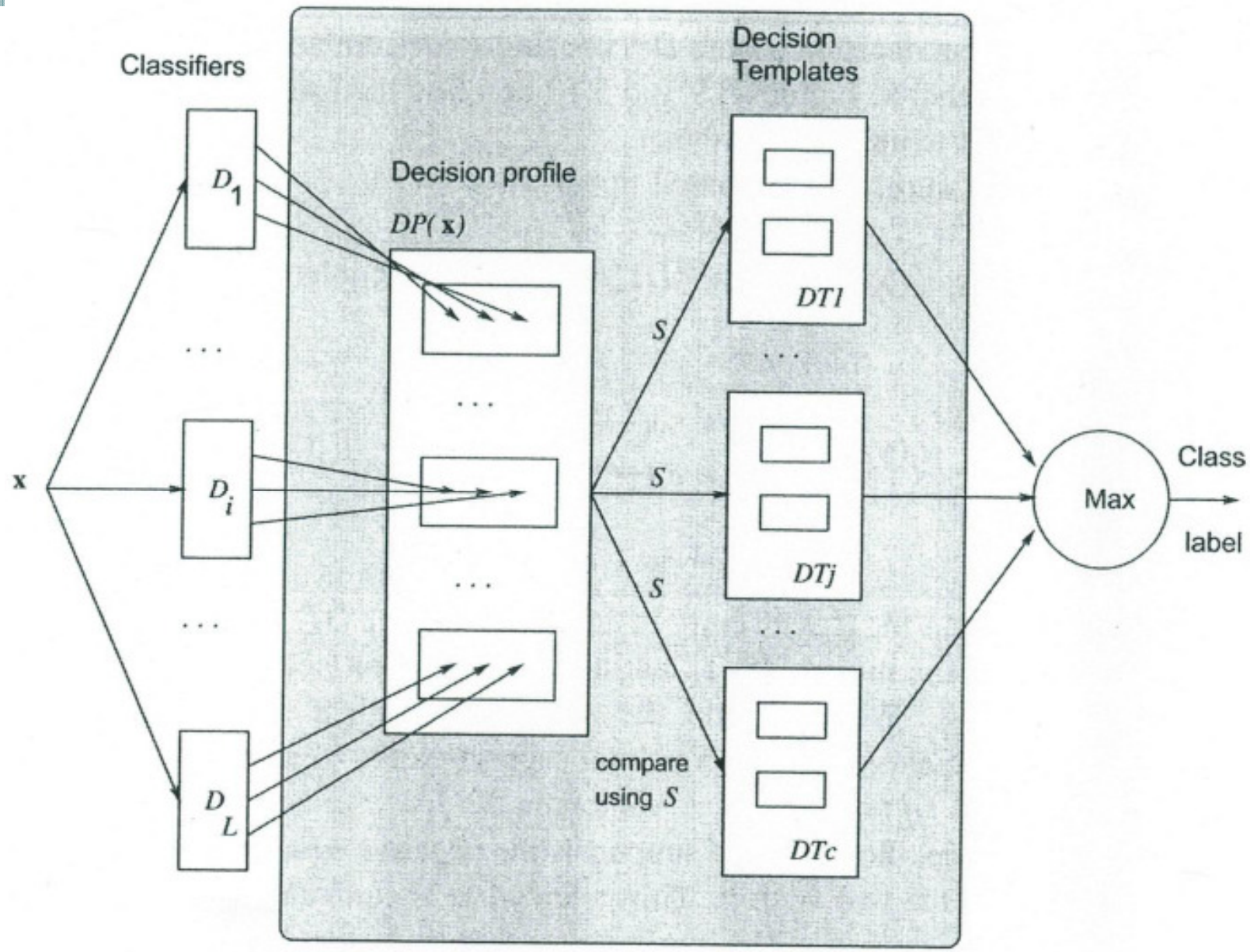
so we have,

$$DT_j = \frac{1}{N_j} \sum_{\substack{\mathbf{z}_k \in \omega_j \\ \mathbf{z}_k \in \mathbf{Z}}} DP(\mathbf{z}_k),$$

where N_j is the number of elements of Z from ω_j

- Operation:
  - Given an input set x € R^n, construct DP(x) and then perform similarity S between DP(x) and each DT_j

$$\mu_j(x) = S(DP(x), DT_j)\ j = 1,\ldots c.$$


Classifiers
Decision Templates
Decision profile
DP(x)
D 1
D i
D L
x
DT1
DTj
DTc
S
S
S
compare
using S
Max
Class
label

Decision Template cont.

- Similarity Measure

  1) Squared Euclidean Distance *(DT(E))*

$$\mu_j(\mathbf{x}) = 1 - \frac{1}{L \times c} \sum_{i=1}^{L} \sum_{k=1}^{c} [DT_j(i,k) - d_{i,k}(\mathbf{x})]^2$$

  where $DT_j(i,k)$ is the (i,k) entry in decision table DT_j

  - Similar to nearest mean classification in intermediate space
  - we can use other distance measures like Minkowski, Mahalanobis etc.

Decision Template cont.

Similarity Measure

2) Symmetric Difference (DT(S))

- This measure comes from fuzzy set theory

$$\mu_j(\mathbf{x}) = 1 - \frac{1}{L \times c} \sum_{i=1}^{L} \sum_{k=1}^{c} \max \{ \min \{ DT_j(i,k), (1 - d_{i,k}(\mathbf{x})) \},$$

$$\min \{ (1 - DT_j(i,k)), d_{i,k}(\mathbf{x}) \} \}$$

Decision Template cont.

- Illustration of Decision Template

$$DT_1 = \begin{matrix} 0.6 & 0.4 \\ 0.8 & 0.2 \\ 0.5 & 0.5 \end{matrix} \qquad DT_2 = \begin{matrix} 0.3 & 0.7 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{matrix}$$

$$DP(x) = \begin{matrix} 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.5 & 0.5 \end{matrix}$$

DT version	***$\mu_1(x)$***	***$\mu_2(x)$***	***Label***
DT(E)	0.9567	0.9333	ω_1
DT(S)	0.5333	0.5333	ω_2

Why Class-Indifferent?

- Decision Templates approach is a context-free (free from the nature of classifier)
- Unlike class-conscious combiners which are idempotent by design
- Assume we have L copies of classifier D in the ensemble and the DTs for the two class are,

$$DT_1 = \begin{matrix} 0.55 & 0.45 \\ \dots & \\ 0.55 & 0.45 \end{matrix} \qquad DT2 = \begin{matrix} 0.2 & 0.8 \\ \dots & \\ 0.2 & 0.8 \end{matrix}$$

and if we have decision of D for **x** to be d1 = 0.4 and d2 = 0.6

Why Class-Indifferent?

- Then all class-conscious methods will assign x to class ω_2
- But based on DT(E) we have 2 Euclidean distance as,

$$E_1 = \sqrt{L \times [(0.55 - 0.40)^2 + (0.45 - 0.60)^2]} = \sqrt{0.045\,L}$$

$$E_2 = \sqrt{L \times [(0.2 - 0.40)^2 + (0.8 - 0.60)^2]} = \sqrt{0.080\,L}$$

- Hence x classified as ω_1.Which means that it is possible that true class is ω_1 hence DTs proved to be correct where other combiners including D will be wrong

Dempster-Shafer Combination

- Its just another method of comparing the DTs and the DP of new **x**
- Instead of calculating the similarity between the DT and DP(x), this method measure the proximity of individual classifiers output with those present in the DT

$$\Phi_{j,i}(\mathbf{x}) = \frac{(1 + ||DT_j^i - D_i(\mathbf{x})||^2)^{-1}}{\sum_{k=1}^{c} (1 + ||DT_k^i - D_i(\mathbf{x})||^2)^{-1}}$$

where DT^i_j = ith row of DT_j and $D_i(x)$ = output of D_i

Dempster-Shafer Combination cont.

- Based on this, we calculate for each class j = 1 to c; for each of the classifier belief degrees as:

$$b_j(D_i(\mathbf{x})) = \frac{\Phi_{j,i}(\mathbf{x}) \prod_{k \neq j} (1 - \Phi_{k,i}(\mathbf{x}))}{1 - \Phi_{j,i}(\mathbf{x})[1 - \prod_{k \neq j} (1 - \Phi_{k,i}(\mathbf{x}))]}$$

- And the final degree of support for the given input is given as:

$$\mu_j(\mathbf{x}) = K \prod_{i=1}^{L} b_j(D_i(\mathbf{x})), \qquad j = 1, \ldots, c$$

Dempster-Shafer Combination cont.

- An illustration,

$$DT_1 = \begin{matrix} 0.6 & 0.4 \\ 0.8 & 0.4 \\ 0.5 & 0.5 \end{matrix} \qquad DT_2 = \begin{matrix} 0.3 & 0.7 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{matrix} \qquad DT_3 = \begin{matrix} 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.5 & 0.5 \end{matrix}$$

Then the 3 proximities for each of 3 decision template:

class	$\Phi_{j,1}(x)$	$\Phi_{j,2}(x)$	$\Phi_{j,3}(x)$
ω_1	0.4587	0.5000	0.5690
ω_2	0.5413	0.5000	0.4310

Dempster-Shafer Combination cont.

- We have the belief equation for ω_1

$$b_1(D_i(\mathbf{x})) = \frac{\Phi_{1,i}(\mathbf{x})(1 - \Phi_{2,i}(\mathbf{x}))}{1 - \Phi_{1,i}(\mathbf{x})[1 - (1 - \Phi_{2,i}(\mathbf{x}))]}$$

$$= \frac{\Phi_{1,i}(\mathbf{x})^2}{1 - \Phi_{1,i}(\mathbf{x})(1 - \Phi_{1,i}(\mathbf{x}))}$$

- Similarly we calculate belief for ω_2, and the final degreeof belief each class 1 and 2 are

Class	$\mathbf{b_j(D_1(x))}$	$\mathbf{b_j(D_2(x))}$	$\mathbf{b_j(D_3(x))}$	$\mu_j(x)$
ω_1	0.2799	0.3333	0.4289	0.5558
ω_2	0.3898	0.3333	0.2462	0.4442

Classifier Fusion using DS

Classifier Fusion using Dempster-Shafer theory of evidence to predict Breast Cancer Tumors

- DS theory of belief was applied to fuse breast cancer data obtained from different diagnostic techniques
- Classifiers used were SVM with linear, polynomial, and RBF kernel
- Classifiers gives beliefs for two classes: benign and malignant
- These evidences are then used to reach a final diagnosis using DS belief combination formula.

References

- L. Kuncheva. (2004) Combining Pattern Classifiers, Methods and Algorithms, Wiley. *
- Raza, Mansoor; Gondal, Iqbal; Green, David; Coppel, Ross L.;"*Classifier Fusion Using Dempster-Shafer theory of evidence to Predict Breast Cancer Tumors*", TENCON 2006. 2006 IEEE Region 10 Conference, 14-17 Nov. 2006, pp. 1-4