# Classifier Combination

Kuncheva Ch. 3

# Motivation

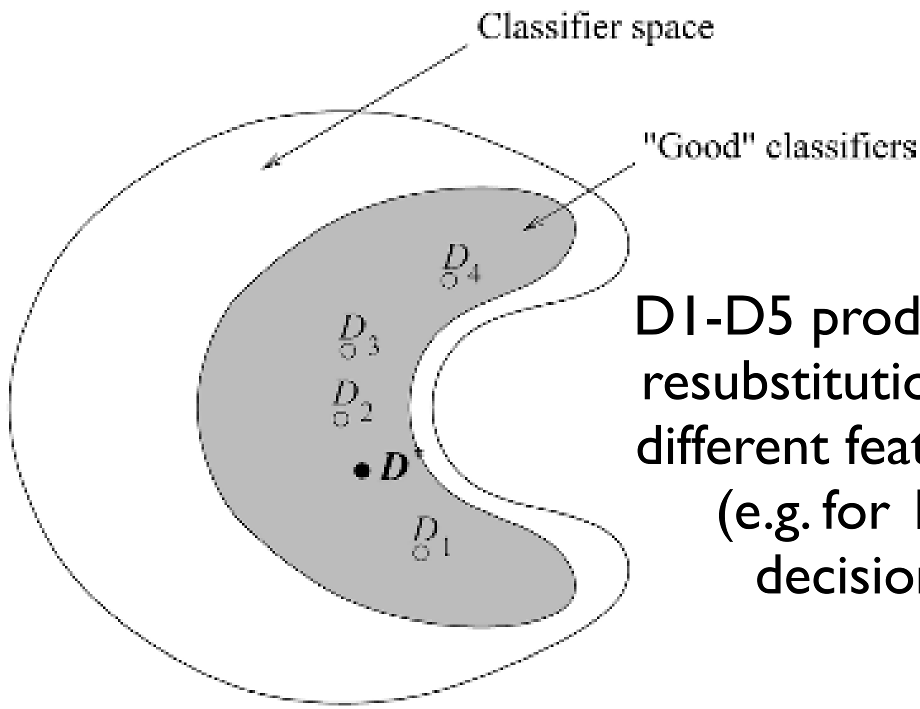
## Classifiers

Are functions that map feature vectors to classes.

- Any single classifier selected will define this function subject to certain biases due to the space of models defined, the training algorithm used, and the training data used.
- Idea: use a 'committee' of base classifiers, map a vector of class outputs or discriminant values for each base classifier to the output classes. (Parallel Combination)
    - Fusion: base classifiers cover feature space; Selection: classifiers are *assigned* to produce classes for a region in feature space
- Another Idea: organize classifiers in a *cascade* (list) or hierarchy, to allow progressively more specialized classifiers to refine intermediate classification decisions, e.g. to classify low-confidence or rejected samples. (Hierarchical/Sequential Combination)

R·I·T

# Effectivene Combinations: Statistical Reason




D1-D5 produced with 0 resubstitution error for different feature subsets (e.g. for 1-NN or decision tree)

"Average" classifier outputs to produce better estimates

***Fig. 3.1*** *The statistical reason for combining classifiers. $D^*$ is the best classifier for the problem, the outer curve shows the space of all classifiers; the shaded area is the space of classifiers with good performances on the data set.*

R·I·T

# Effective Combinations: Computational Reason

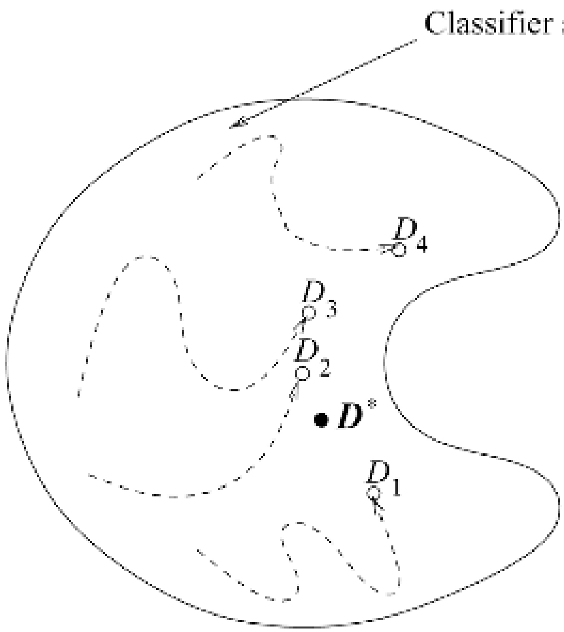



***Fig. 3.2*** *The computational reason for combining classifiers. $D^*$ is the best classifier for the problem, the closed space shows the space of all classifiers, the dashed lines are the hypothetical trajectories for the classifiers during training.*

D1-D4: classifiers trained using hill climbing (e.g. gradient descent), random search

Aggregation of local-search optimizations may improve over individual (local) error minima

# Effective Combinations: Representational Reason

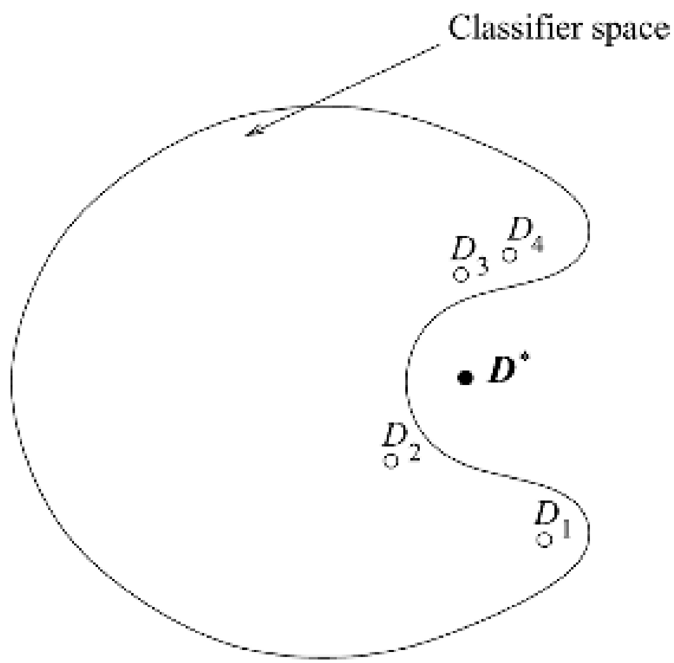



***Fig. 3.3*** *The representational reason for combining classifiers. $D^*$ is the best classifier for the problem; the closed shape shows the chosen space of classifiers.*

D1-D4: four linear classifiers (e.g. SVM with fixed kernel)

Combination allowing decision boundaries not expressible in original classifier parameter space to be represented

R·I·T

# Classifier Output Types
## *(Xu, Krzyzak, Suen)*

### Type 1: Abstract Level

Chosen class label for each base classifier

### Type 2: Rank Level

List of ranked class labels for each base classifier

### Type 3: Measurement Level

Real values (e.g. [0,1]) for each class (discriminant function outputs)

# Fusion Combinations (*k* base classifiers)

## For Type 1 (Single Label per Base Classifier)

$$D_l(x) = C(B(x))$$
$$B : \mathbb{R}^n \rightarrow \Omega^k, \quad C : \Omega^k \rightarrow \Omega$$

*Combination example: voting*

*Intermediate Feature Space*

## For Type 2 (Ranked list of *r* classes)

$$D_r(x) = C(B(x))$$
$$B : \mathbb{R}^n \rightarrow \Omega^{rk}, \quad C : \Omega^{rk} \rightarrow \Omega$$

*Combination example: weighted voting (e.g. Borda Count)*

## For Type 3 (Discriminant Values)

$$D_m(x) = C(B(x))$$
$$B : \mathbb{R}^n \rightarrow \mathbb{R}^{|\Omega|k}, \quad C : \mathbb{R}^{|\Omega|k} \rightarrow \Omega$$

*Combination example: min, max, product rules*

R·I·T

# Classifier Ensembles (Fusion): Combination Techniques

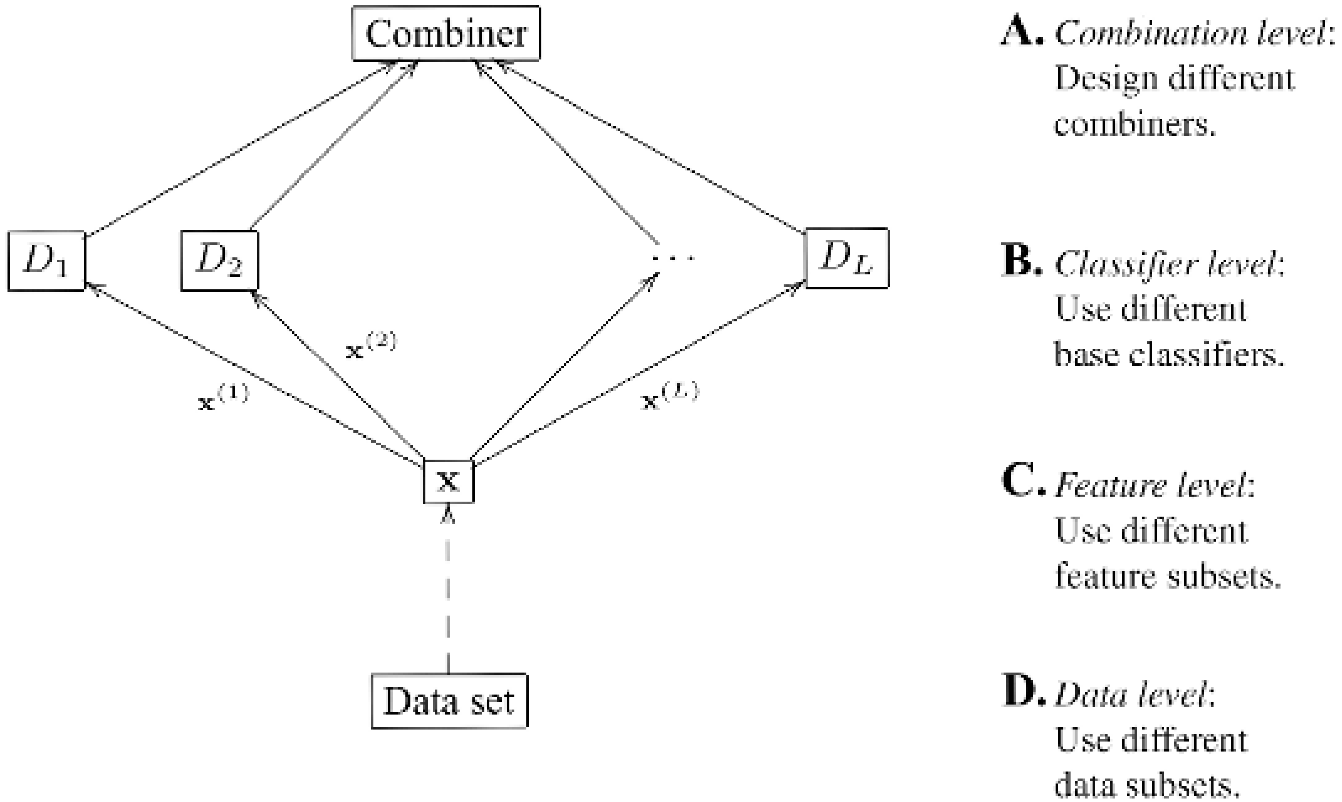
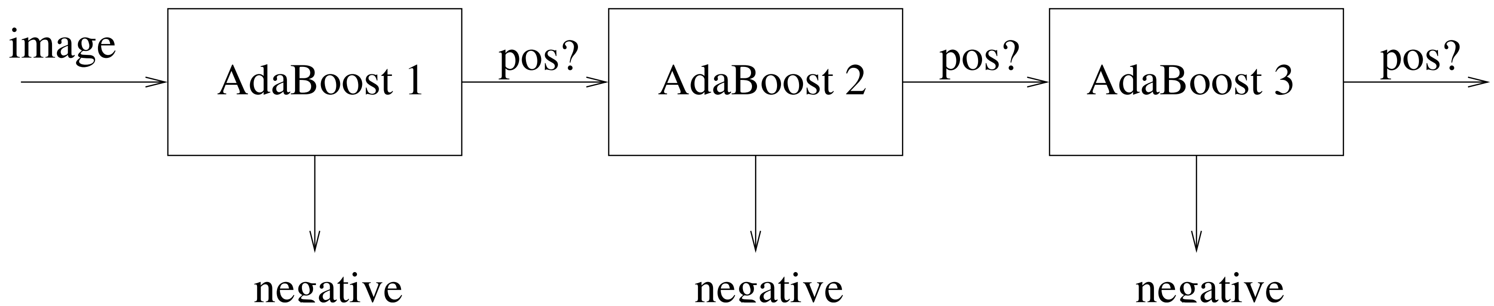



***Fig. 3.4*** *Approaches to building classifier ensembles.*

R·I·T

# Cascade Architecture

*McCane, Novins and Albert, "Optimizing Cascade Classifiers" (unpublished, 2005)




Here a set of classifiers is obtained using AdaBoost, then partitioned using dynamic programming to produce a cascade of binary classifiers (detectors)

(Viola and Jones face detector (2001, 2004))

# ECOC: Another Label-Based Combiner

## Error-Correcting Output Codes (ECOC)

Classifier ensemble comprised of binary classifiers (each distinguishing a *subset* of the class labels: *dichotomizers*)

- Represent base classifier output as a bit string
- Learn/associate bit string sequences with concrete labels; classify by Hamming distance to bit string ('code') for each class $$\mu_j(\mathbf{x}) = -\sum_{i=1}^{L} |s_i - C(j, i)|$$ *('support'/discriminant value)*
- Details provided in Ch. 8 (Kuncheva)

$(s_1, \ldots, s_L) = (0, 1, 1, 0, 1, 0, 1).$

Hamming Distances:
class 1: 5
class 2: 3
class 3: 1
class 4: 5

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|
| $\omega_1$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| $\omega_2$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $\omega_3$ | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| $\omega_4$ | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

R·I·T

# Training Combiners: Stacked Generalization

| | | | | |
|---|---|---|---|---|
| Training | A | B | C | D |
| | B | C | D | A |
| | C | D | A | B |
| Testing | D | A | B | C |

***Fig. 3.5*** *Standard four-fold cross-validation set-up.*

## Protocol:

Train base classifiers using cross-fold validation

Then train combiner on all N points by using the class labels output by the base classifiers for each fold (train/test partition)

# There is nothing new under the sun...

Sebeysten (1962)

Idea of using classifier outputs as input features, classifier cascade architectures

Dasarthy and Sheila (1975)

Classifier selection using two classifiers

Rastrigin and Erenstein (1981 - Russian)

Dynamic classifier selection

Barabash (1983)

Theoretical results on majority vote classifier combination