R·I·T

Pattern Recognition: An Overview

Prof. Richard Zanibbi

Pattern Recognition

(One) Definition

The identification of implicit objects, types or relationships in raw data by an animal or machine

- i.e. *recognizing* hidden information in data

Common Problems

- What is it?
- Where is it?
- How is it constructed?
- These problems interact. Example: in optical character recognition (OCR), detected characters may influence the detection of words and text lines, and vice versa

R·I·T

Pattern Recognition: Common Tasks

What is it? (Task: Classification)

Identifying a handwritten character, CAPTCHAs; discriminating humans from computers

Where is it? (Task: Segmentation)

Detecting text or face regions in images

How is it constructed? (Tasks: Parsing, *Syntactic* Pattern Recognition)

Determining how a group of math symbols are related, and how they form an expression;

Determining protein structure to decide its type (class) (an example of what is often called "Syntactic PR")

Models and Search: Key Elements of Solutions to Pattern Recognition Problems

Models

For algorithmic solutions, we use a formal model of entities to be detected. This model represents knowledge about the problem domain ('prior knowledge'). It also defines the space of possible inputs and outputs.

Search: Machine Learning and Finding Solutions

Normally model parameters set using "learning" algorithms

- Classification: learn parameters for function from model inputs to classes
- Segmentation: learn search algorithm parameters for detecting Regions of Interest (ROIs: note that this requires a classifier to identify ROIs)
- Parsing: learn search algorithm parameters for constructing structural descriptions (trees/graphs, often use sementers & classifiers to identify ROIs and their relationships in descriptions)

Major Topics

Topics to be covered this quarter:

Bayesian Decision Theory

Feature Extraction

*Various Classification Models

Classifier Combination

Clustering (segmenting data to define classes)

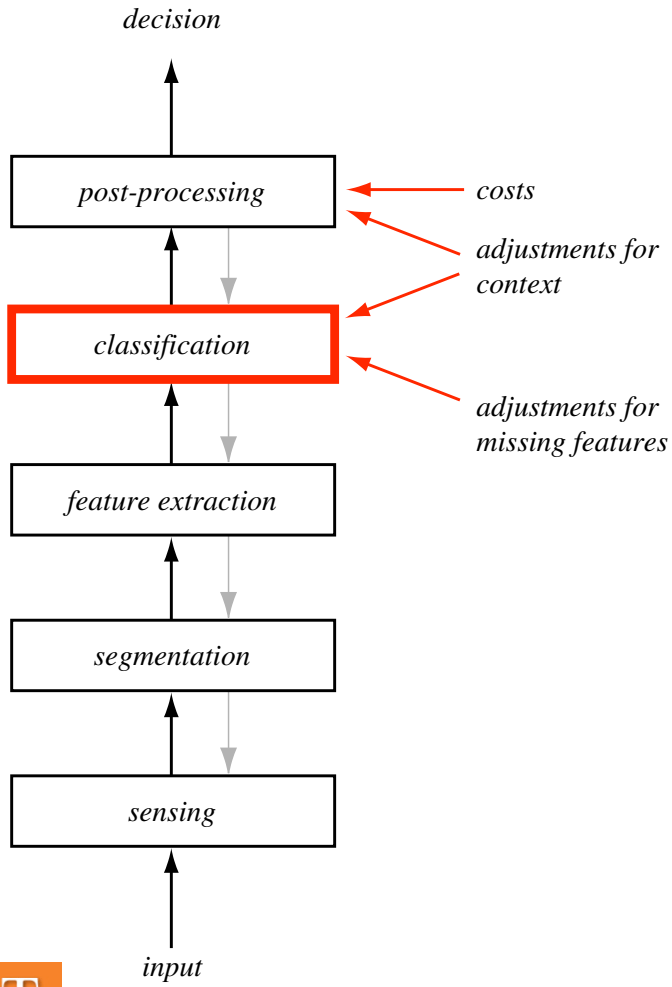
Syntactic Pattern Recognition

R·I·T

Pattern Classification

(Overview)

Classifying an Object



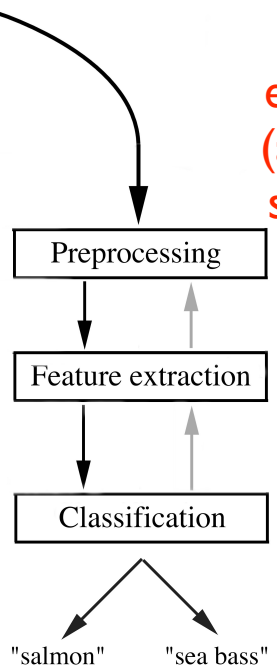
Obtaining Model Inputs

Physical signals converted to digital signal (transducer(s)); a region of interest is identified, features computed for this region

Making a Decision

Classifier returns a class; may be revised in post-processing (e.g. modify recognized character based on surrounding characters)

Example (DHS): Classifying Salmon and Sea Bass



e.g. image processing
(adjusting brightness)
segment fish regions

FIGURE 1.1. The objects to be classified are first sensed by a transducer (camera), whose signals are preprocessed. Next the features are extracted and finally the clas-

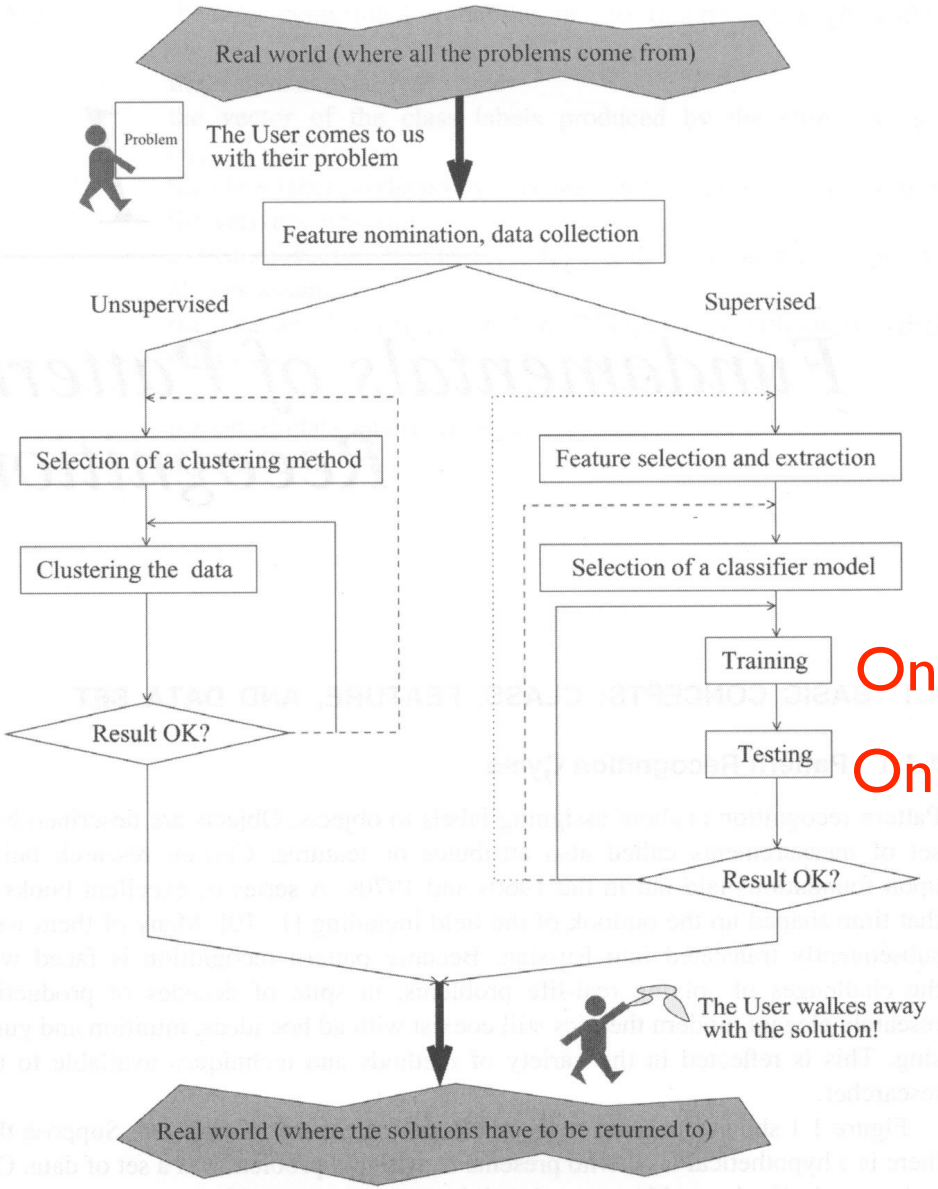


Fig. 1.1 *The pattern recognition cycle.*

Designing a classifier or clustering algorithm

On a training set (learn parameters)

On a *separate* testing set

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

Feature Selection and Extraction

Feature Selection

Choosing from available features those to be used in our classification model. Ideally, these:

- Discriminate well between classes
- Are simple and efficient to compute

Feature Extraction

Computing features for inputs at run-time

Preprocessing

User to reduce data complexity and/or variation, and applied before feature extraction to permit/simplify feature computations; sometimes involves other PR algorithms (e.g. segmentation)

Types of Features

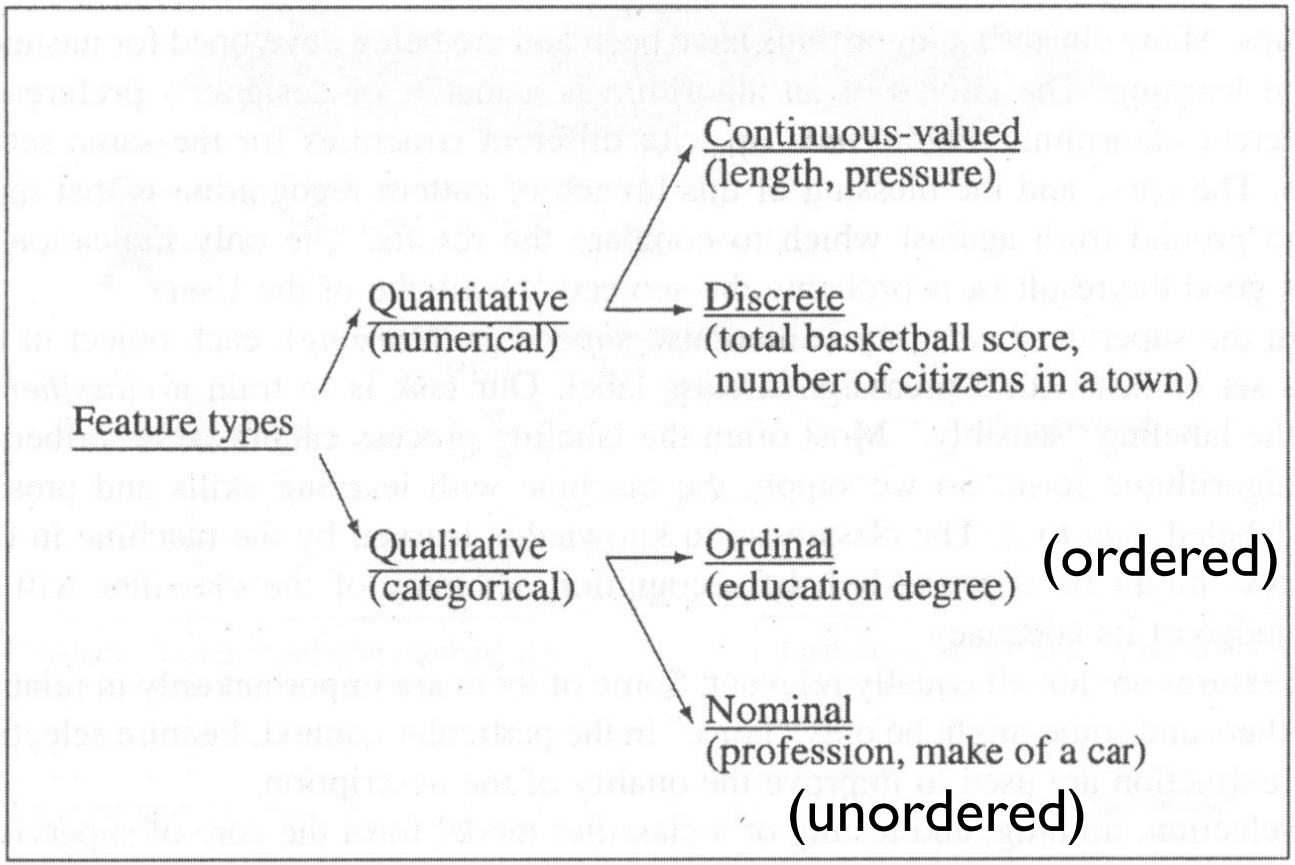
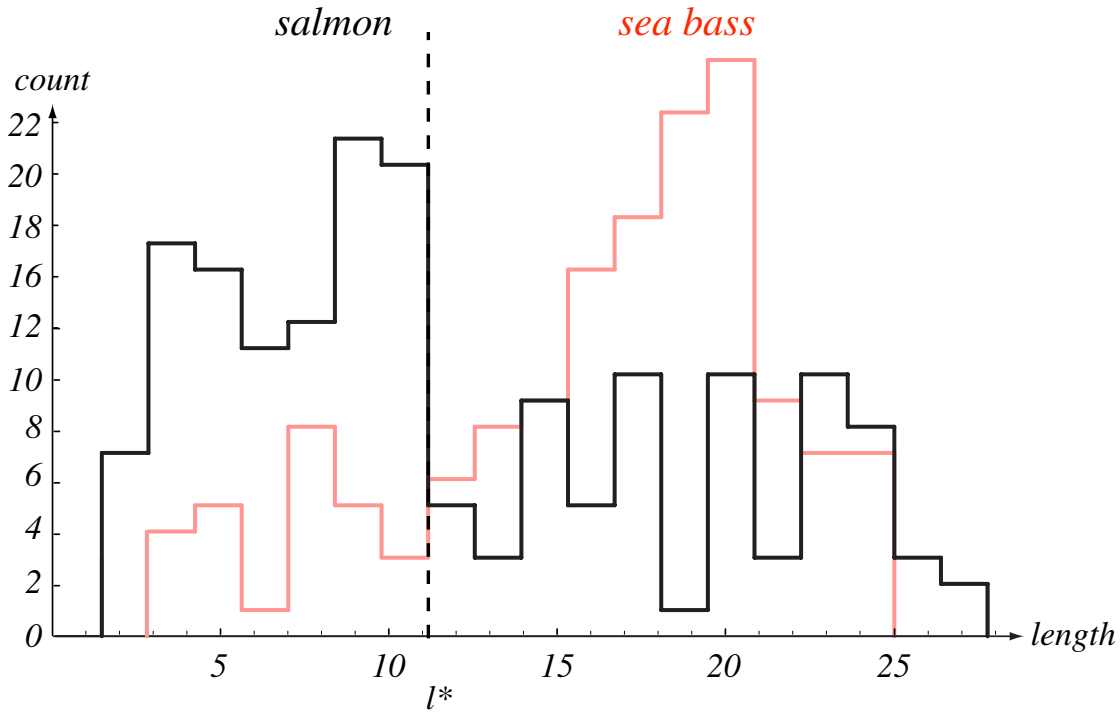


Fig. 1.2 *Types of features.*

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

Example Single Feature (DHS): Fish Length



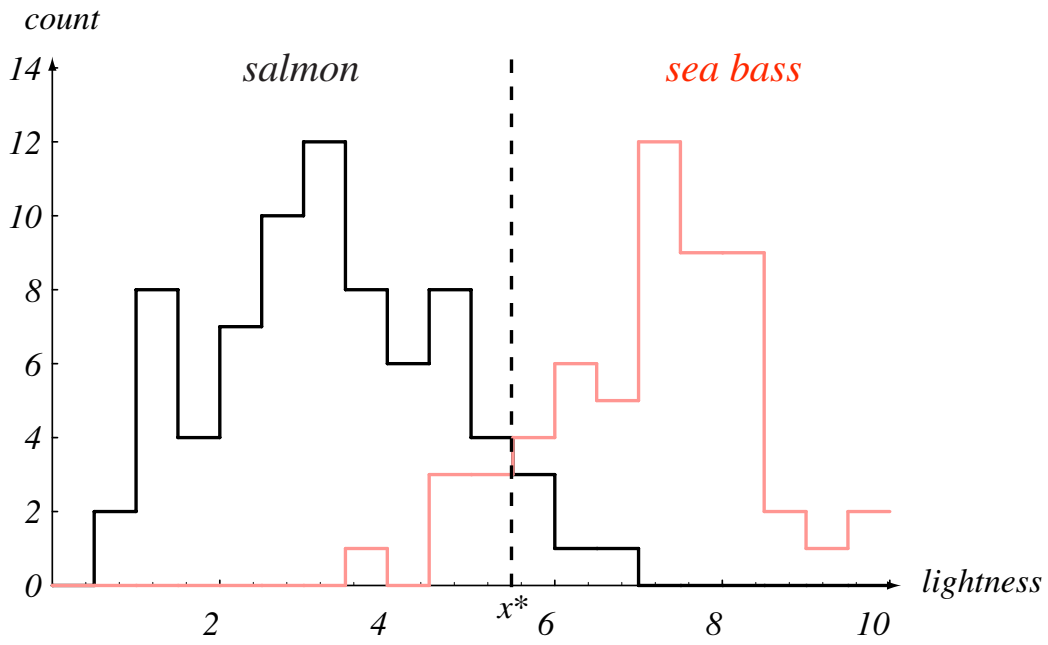
A Poor Feature for Classification

Computed on a training set

No threshold will prevent errors

Threshold *l** shown will produce fewest errors on average

A Better Feature: Average Lightness of Fish Scales



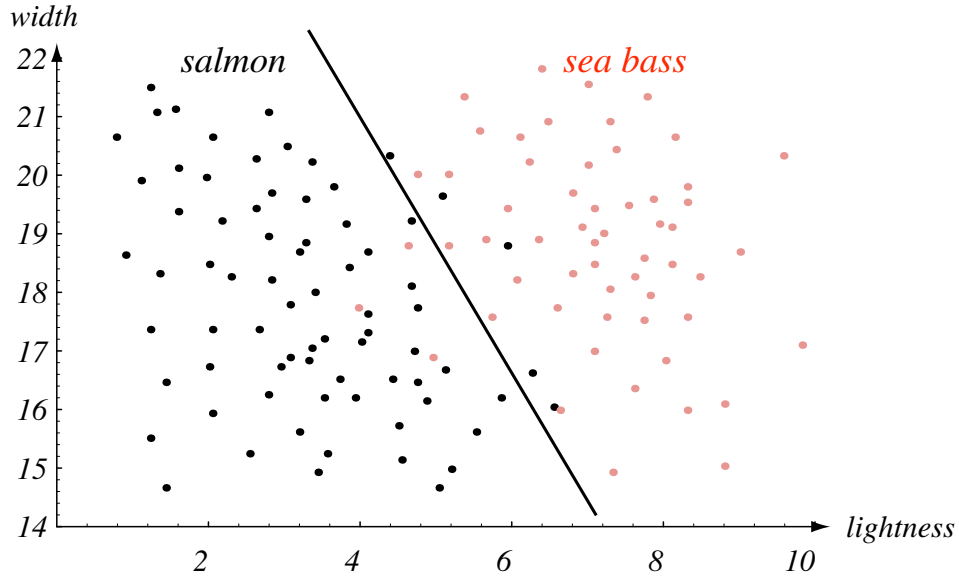
Still some errors

even for the best threshold, x^* (again, min. average # errors)

Unequal Error Costs

If worse to confuse bass for salmon than vice versa, we can move x^* to the left

A Combination of Features: Lightness and Width



In general, determining appropriate features is a difficult problem, and determining optimal features is often impractical or impossible.

Feature Space

Is now two-dimensional; fish described in model input by a *feature vector* (x1, x2) representing a point in this space

Decision Boundary

A linear discriminant (line used to separate classes) is shown; still some errors

Classifier: A Formal Definition

Classifier (continuous, real-valued features)

Defined by a function from a n-dimensional space of real numbers to a set of *c* classes, i.e.

$$D : R^n \rightarrow \Omega, \text{ where } \Omega = \{\omega_1, \omega_2, \ldots \omega_c\}$$

Canonical Model

Classifier defined by *c* discriminant functions, one per class. Each returns a real-valued "score." Classifier returns the class with the highest score.

$$g_i : R^n \rightarrow R, \quad i = 1, \ldots c$$
$$D(x) = \omega_{i*} \in \Omega \iff g_{i*} = \max_{i=1,\ldots,c} g_i(x)$$

R·I·T

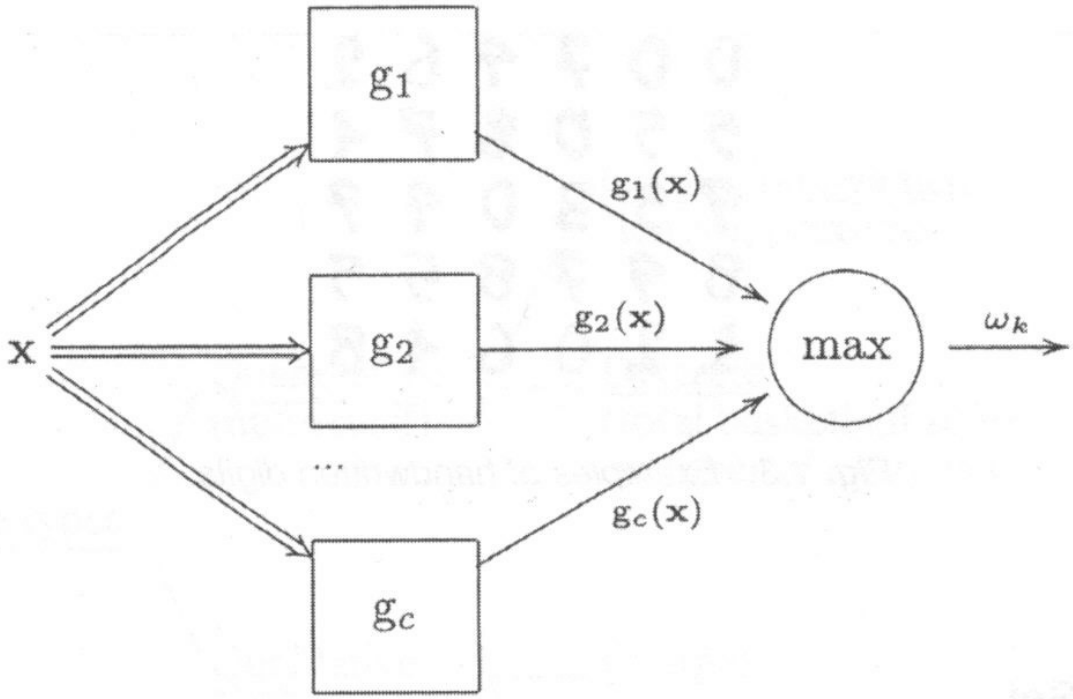


Fig. 1.4 *Canonical model of a classifier. The double arrows denote the n-dimensional input vector* ***x****, the output of the boxes are the discriminant function values, $g_i(x)$ (scalars), and the output of the maximum selector is the class label $\omega_k \in \Omega$ assigned according to the maximum membership rule.*

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

Regions and Boundaries

Classification (or Decision) Regions

Regions in feature space where one class has the highest discriminant function "score"

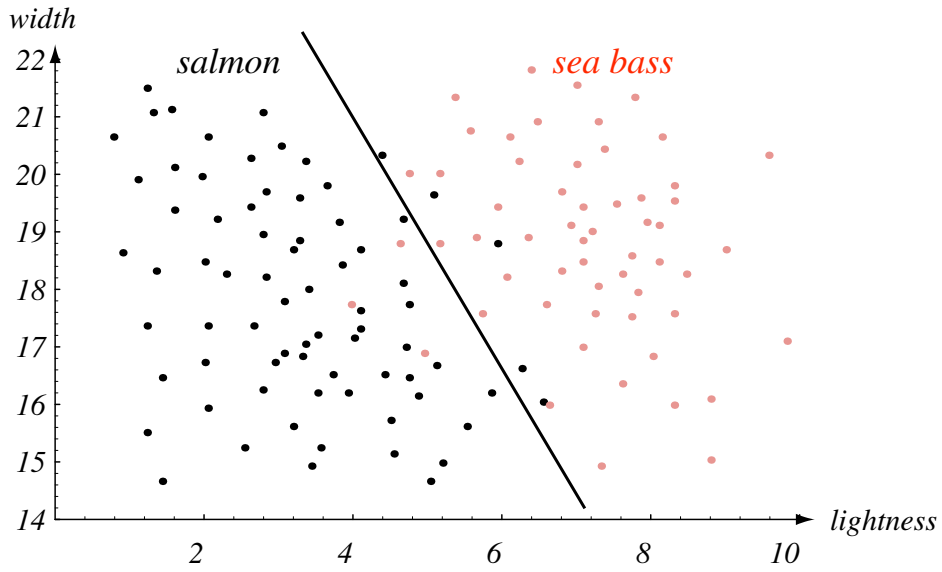
$$R_i = \left\{ x \middle| x \in R^n, \quad g_i(x) = \max_{k=1,\ldots,c} g_k(x) \right\}, \quad i = 1, \ldots, c$$

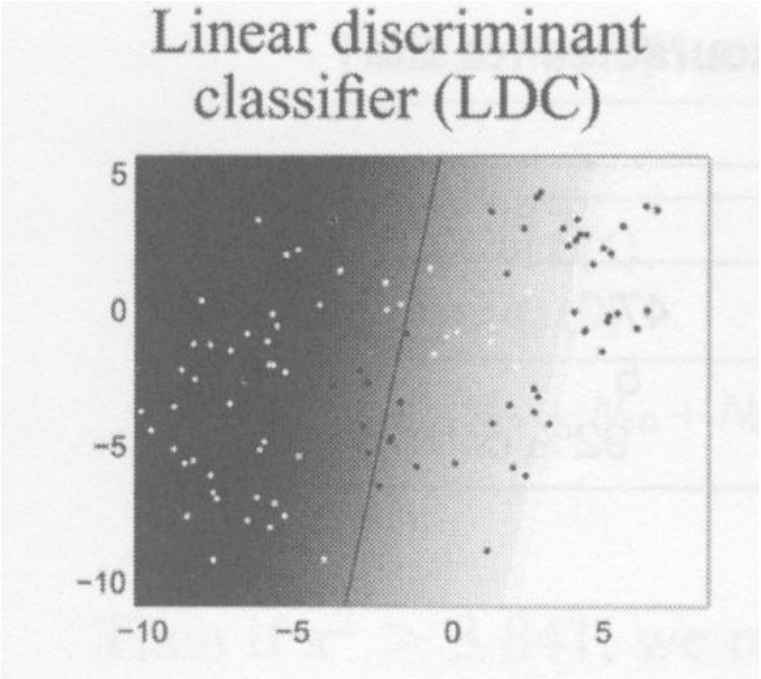
Classification (or Decision) Boundaries

Exist where there is a tie for the highest discriminant function value

R·I·T

Example: Linear Discriminant Separating Two Classes





(from Kuncheva: visualizes changes (gradient) for class score)

"Generative" Models

"Discriminative" Models

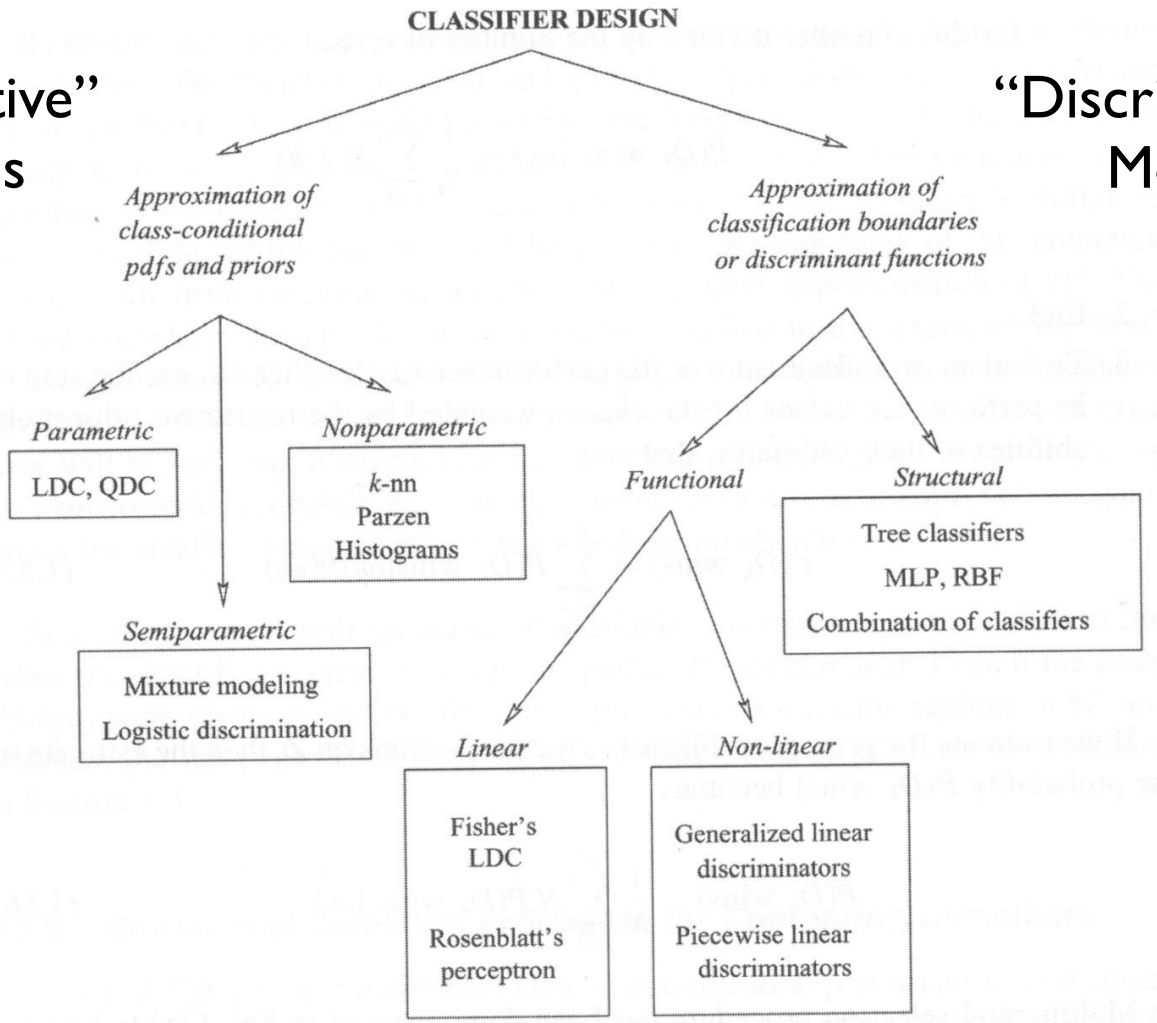


Fig. 1.13 *A taxonomy of methods for classifier design.*

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

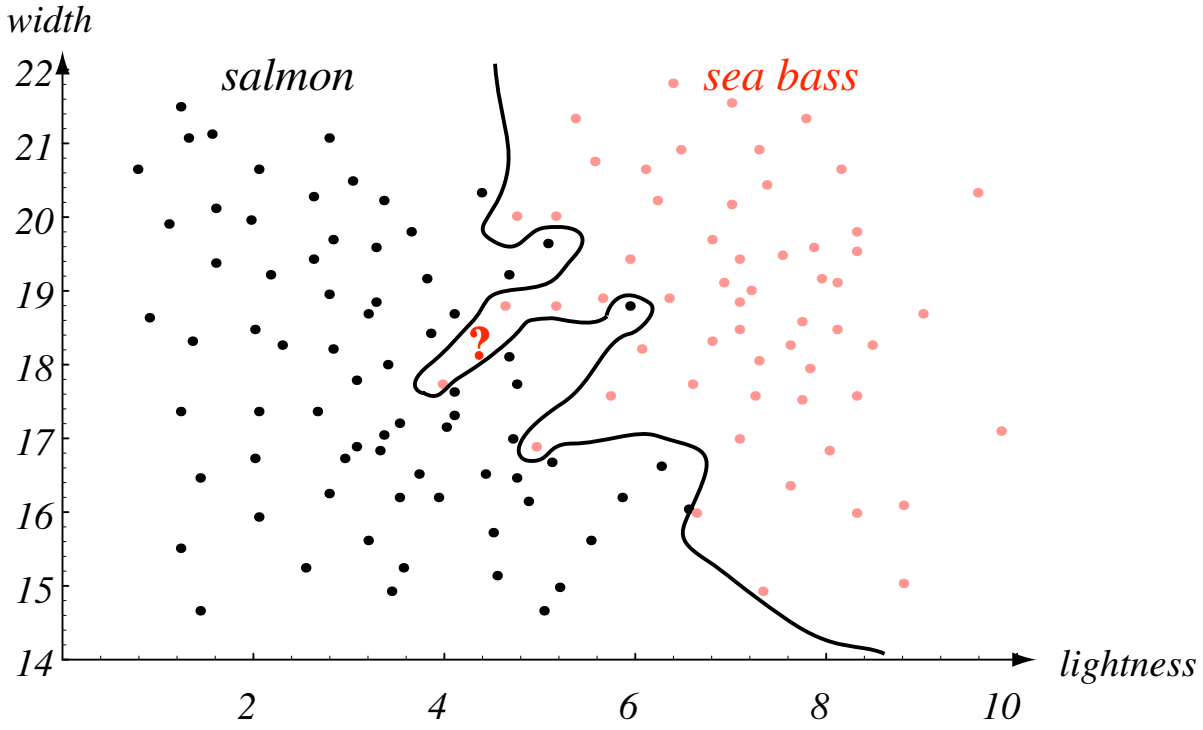
Generalization

Too Much of A Good Thing

If we build a "perfect" decision boundary for our training data, we will produce a classifier making no errors on the training set, but performing poorly on unseen data

- i.e. the decision boundary does not "generalize" well to the true input space, and new samples as a result

Poor Generalization due to Over-fitting the Decision Boundary



Question-?

Marks a salmon that will be classified as a sea bass.

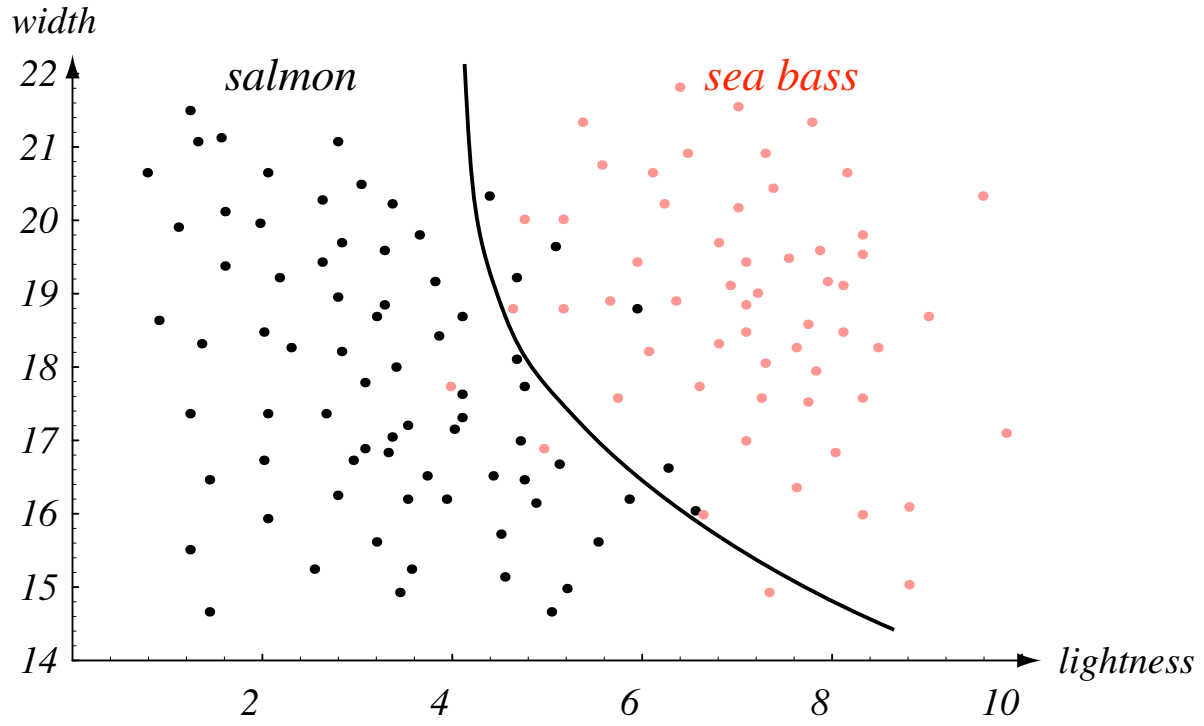
Avoiding Over-Fitting

A Trade-off

We may need to accept more errors on our training set to produce fewer errors on new data

- We have to do this *without "peeking at"* (repeatedly evaluating) the test set, otherwise we over-fit the test set instead
- Occam's razor: prefer simpler explanations over those that are unnecessarily complex
- Practice: simpler models with fewer parameters are easier to learn/more likely to converge. A poorly trained "sophisticated model" is often of no use.

A Simpler Decision Boundary, with Better Generalization



R·I·T

"No Free Lunch"

One size does not fit all

Because of great differences in the structure of feature spaces, the structure of decision boundaries between classes, error costs, and differences in how classifiers are used to support decisions, creating a single *general purpose* classifier is "profoundly difficult" (DHS) - maybe impossible?

Put another way...

There is *no* "best classification model," as different problems have different requirements

R·I·T

Clustering

(trying to *discover* classes in data)

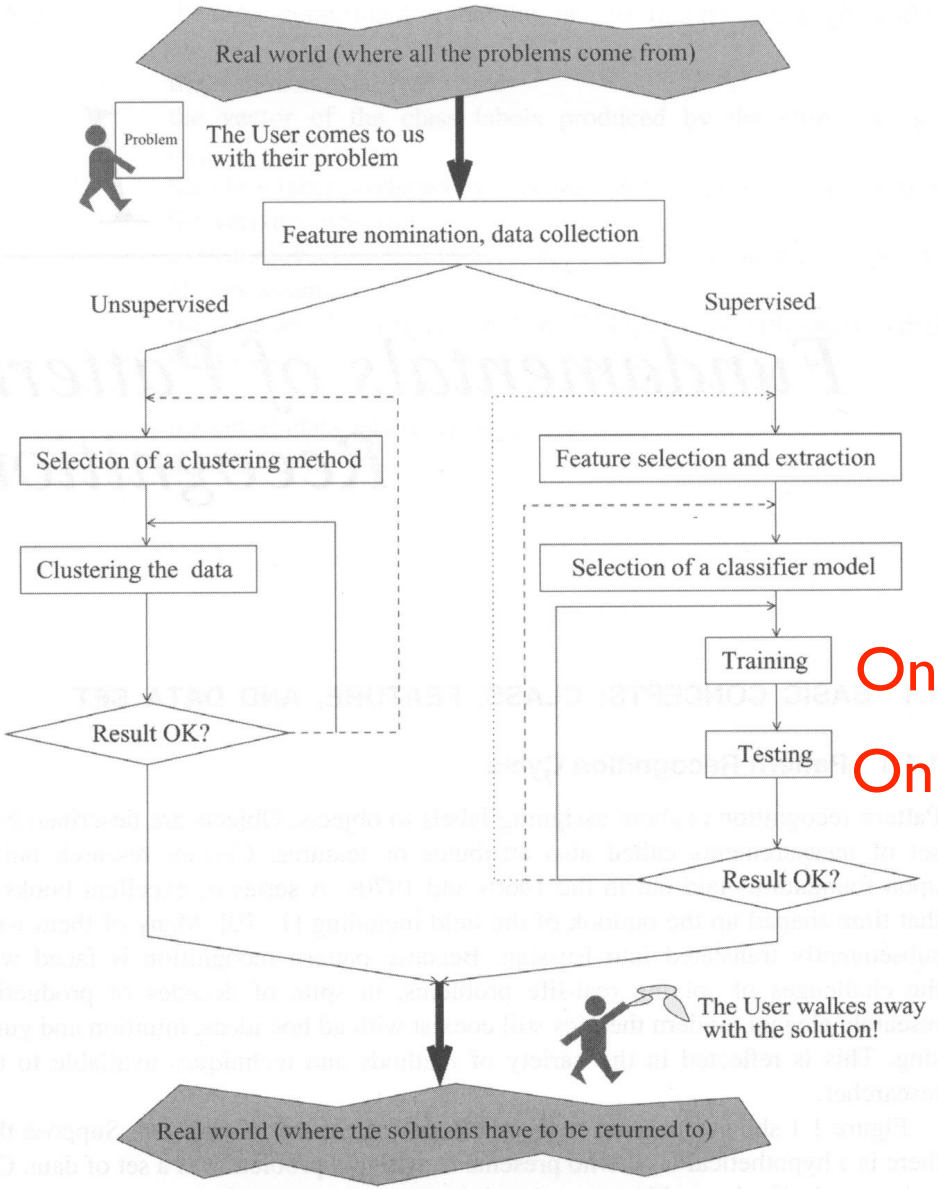


Fig. 1.1 *The pattern recognition cycle.*

Designing a classifier or clustering algorithm

On a training set (learn parameters)

On a *separate* testing set

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

Clustering

The Task

Given unlabeled data set Z, partition the data points into disjoint sets ("clusters:" each data point is included in exactly one cluster)

Main Questions Studied for Clustering:

- Is there structure in the data, or does our clustering algorithm *impose* structure?
- How many clusters should we look for?
- How do we define object similarity in feature space?
- How do we know when clustering results are "good"?

R·I·T

Hierarchical: constructed by merging most similar clusters at each iteration

Single linkage clustering

1. Pick the number of clusters c and a similarity measure $\mathcal{S}(a, b)$ between two objects a and b. Initialize the procedure by defining an individual cluster for each point in $\mathbf{Z}$.
2. Identify the two most similar clusters and join them as a new cluster, dismissing the initial two clusters. The similarity between clusters A and B is measured as
$$\min_{a \in A, b \in B} \mathcal{S}(a, b).$$
3. Repeat step 2 until c clusters are found.

Fig. 1.14 *The single linkage clustering algorithm.*

Non-Hierarchical: all points assigned to a cluster each iteration

***c*-Means clustering**

1. Pick the number of clusters c and a similarity measure $\mathcal{S}(a, b)$ between two objects a and b. Initialize the c cluster centers (e.g., by randomly selecting c points from $\mathbf{Z}$ to be the centers).
2. Label all points in $\mathbf{Z}$ with respect to their similarity to the cluster centers: each point is assigned to the cluster with the most similar center.
3. Calculate the new centers as the mean of the points from $\mathbf{Z}$ assigned to the respective cluster.
4. Repeat steps 2 and 3 until no change in the centers occurs.

Fig. 1.15 *The c-means clustering algorithm.*

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004

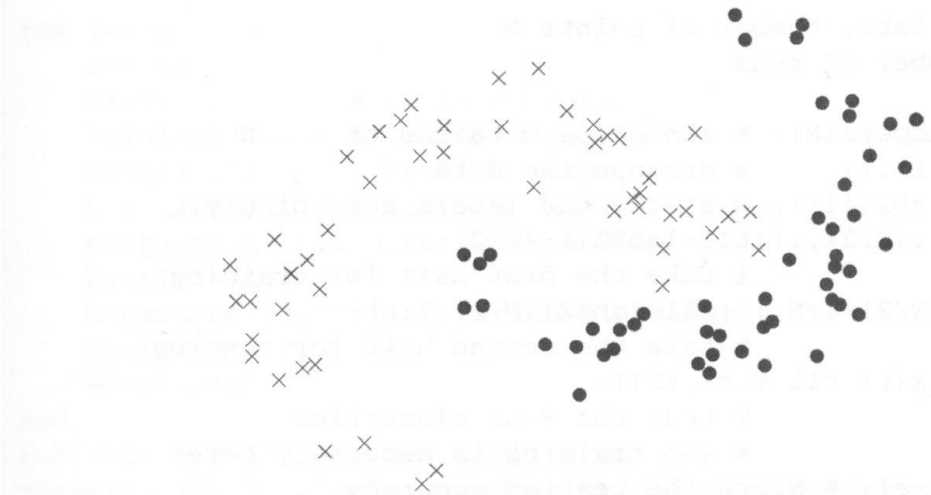


Fig. 1.16 *Banana data for the clustering example.*

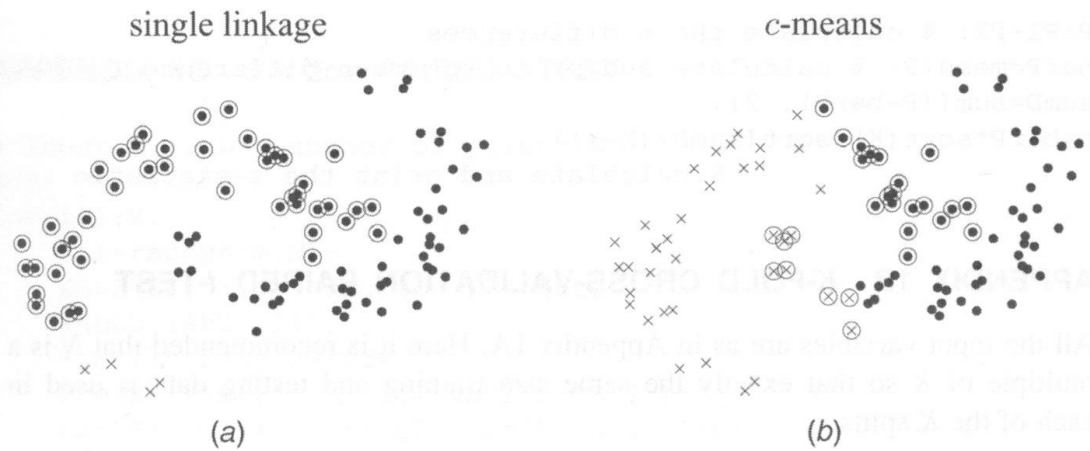


Fig. 1.17 *Results from the single linkage (a) and c-means (b) clustering on a banana data set with 50 points on each banana shape. The "mislabeled" points are encircled.*

from "Combining Pattern Classifiers" by L. Kuncheva, Wiley, 2004