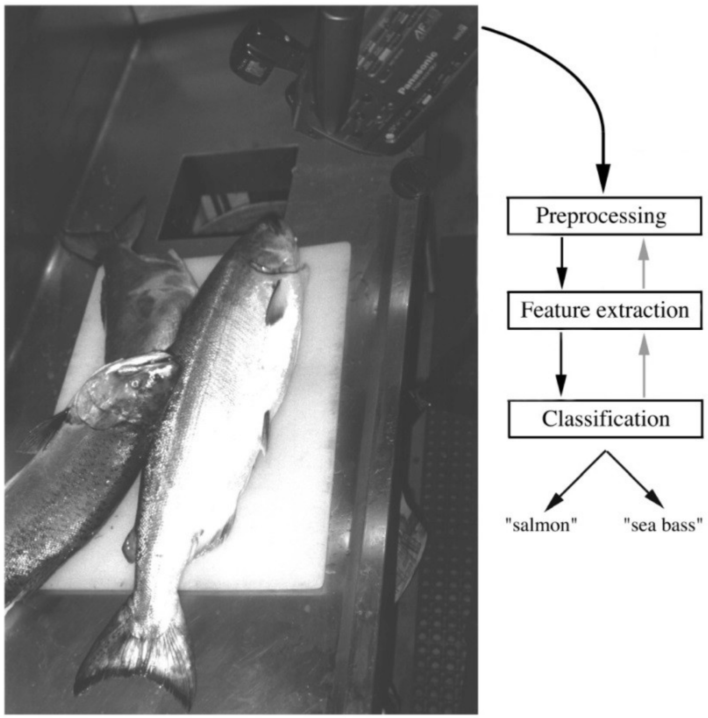
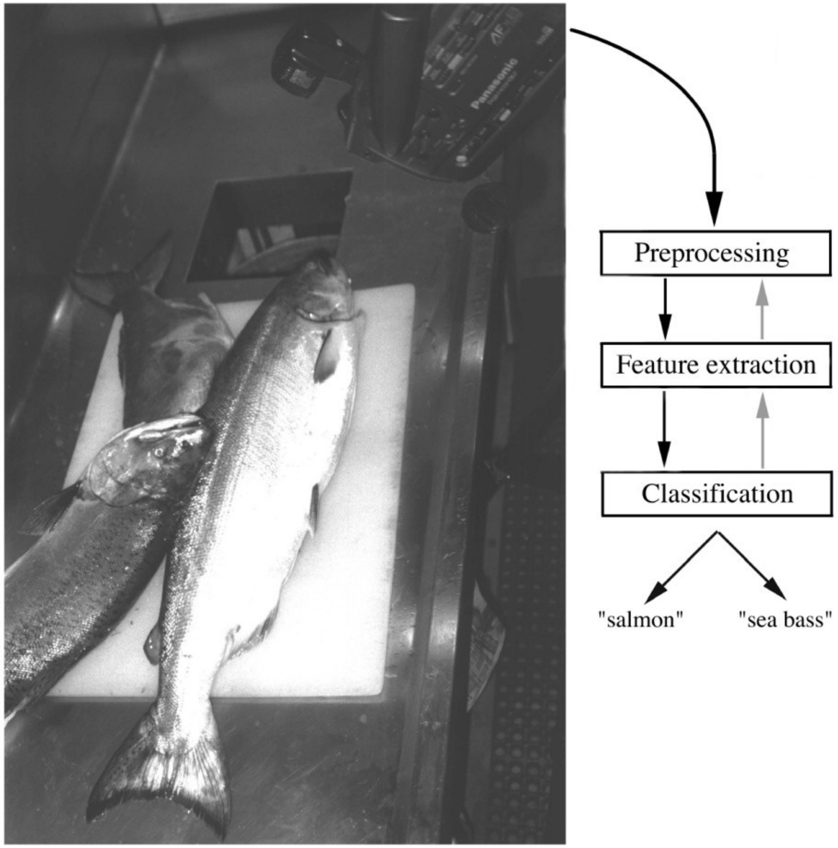
Complexity of PR – An Example

Problem: Sorting incoming fish on a conveyor belt.

Assumption: Two kind of fish:
(1) sea bass
(2) salmon



1. Pre-processing Step



Example

(1) Image enhancement

(2) Separate touching or occluding fish

(3) Find the boundary of each fish

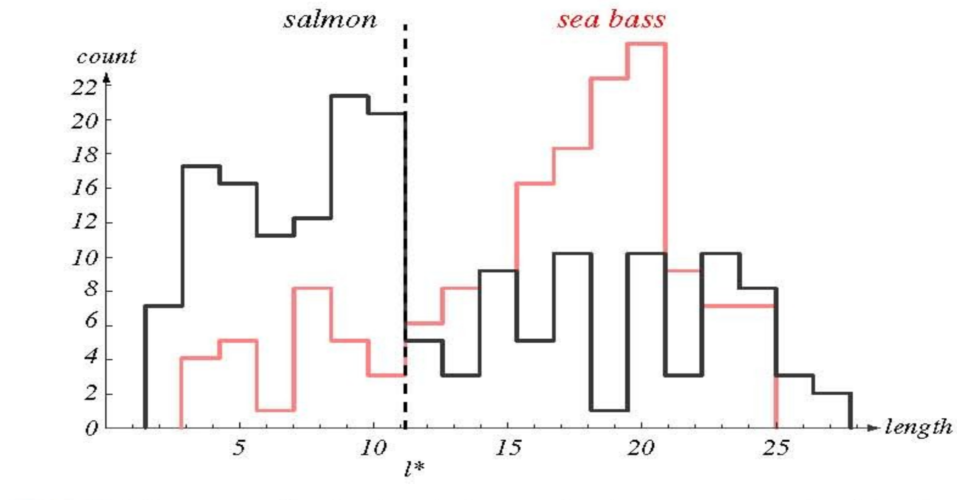
2. Feature Extraction

- Assume a fisherman told us that a sea bass is generally longer than a salmon.

- We can use length as a feature and decide between sea bass and salmon according to a threshold on length.

- How should we choose the threshold?

“Length” Histograms

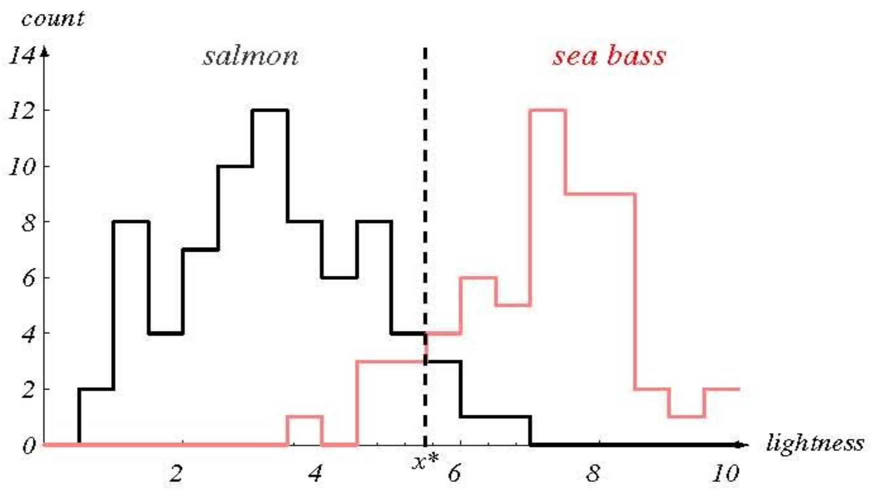


threshold *l**

- Even though sea bass is longer than salmon on the average, there are many examples of fish where this observation does not hold.

"Average Lightness" Histograms

- Consider a different feature such as "average lightness"



threshold x^*

- It seems easier to choose the threshold x^* but we still cannot make a perfect decision.

Multiple Features

- To improve recognition accuracy, we might have to use more than one features at a time.
  - Single features might not yield the best performance.
  - Using combinations of features might yield better performance.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- x_1: length
- x_2: lightness

- How many features should we choose?

Feature Extraction

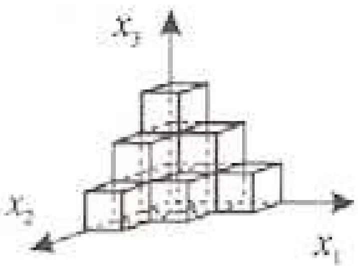
It might be difficult and computationally expensive to extract certain features.

Challenges in Feature Extraction

i. How many features?
ii. How to choose a good set of features?
iii. How to handle the missing features?

i) How Many Features?

- Adding too many features can, paradoxically, lead to a worsening of performance.
  - Divide each of the input features into a number of intervals, so that the value of a feature can be specified approximately by saying in which interval it lies.



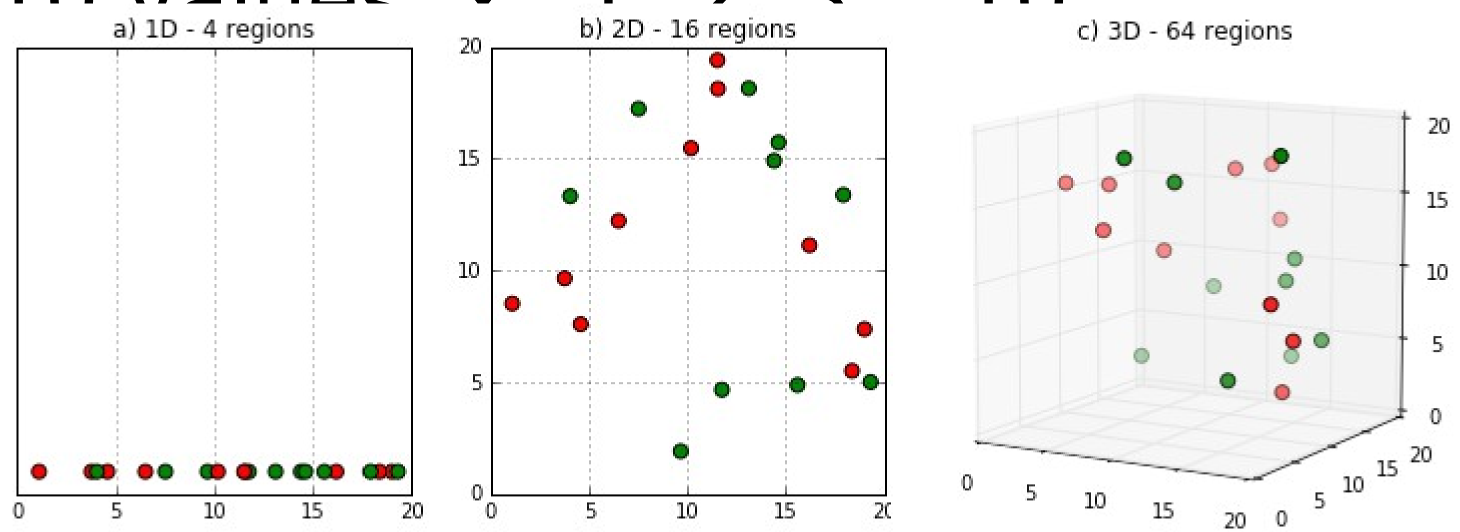
  - If each input feature is divided into **M** divisions, then the total number of cells is $\mathbf{M}^d$ (***d:*** # of features).
  - Since each cell must contain at least one point, the number of training data grows exponentially with ***d*** also known as curse of dimensionality.

Curse of Dimensionality- definition

- As the number of features or dimensions grows, **the amount of data we need to generalize accurately grows exponentially."**
- In applied maths, COD refers to the **problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space.**

Curse of Dimensionality-(contd)

- Fig. 1 (a) shows 10 data points in one dimension i.e. there is only one feature in the data set. It can be easily represented on a line with only 10 values, x=1, 2, 3... 10.

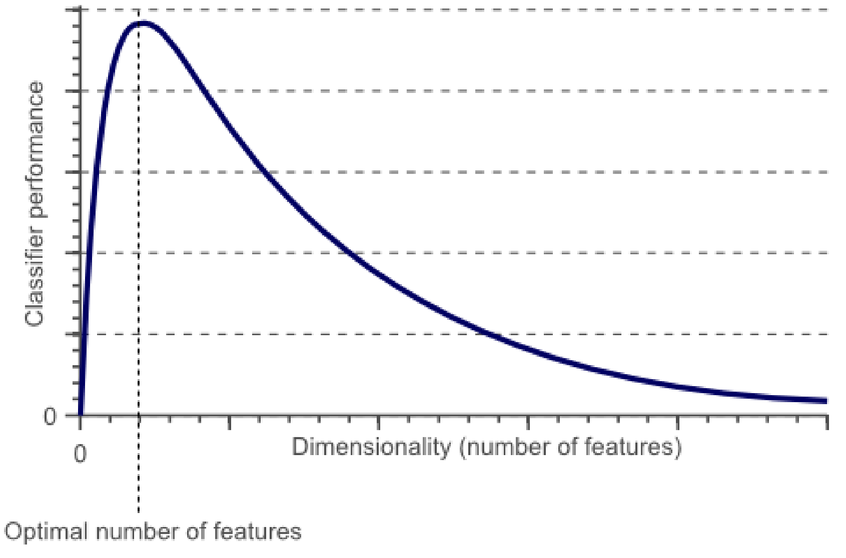


Curse of Dimensionality- (contd)

- But if we add one more feature, same data will be represented in 2 dimensions (Fig.1 (b)) causing increase in dimension space to 10*10 =100.
- And again if we add 3rd feature, dimension space will increase to 10*10*10 = 1000. As dimensions grows, dimensions space increases exponentially.
- 10^1 = 10
- 10^2 = 100
- 10^3 = 1000 and so on...

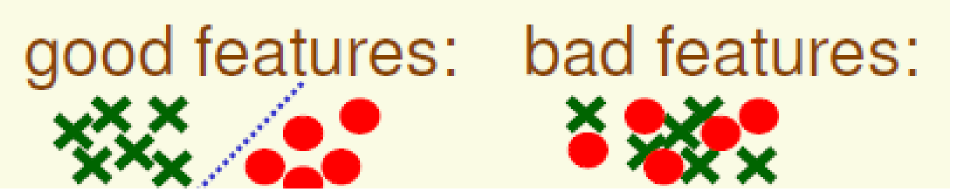
Does adding more features always improve performance?

- No.



ii) How to choose a good set of features?

- How to choose a good set of features?
  - Discriminative features



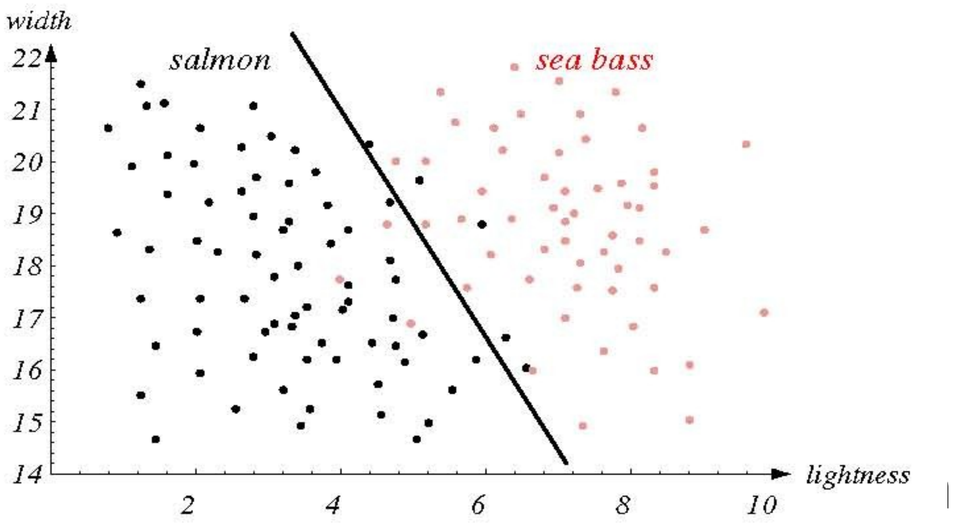
  - Invariant features (e.g., translation, rotation and scale)
- Correlated features might not improve performance.
- Are there ways to automatically learn which features are best ?

iii) Missing Features

- Certain features might be missing (e.g., due to occlusion).
- How should we train the classifier with missing features ?
- How should the classifier make the best decision with missing features ?

3. Classification

- Partition the *feature space* into two regions by finding the decision boundary that minimizes the error.



- How sho
boundary?

Main Classification Approaches

x: *input vector (pattern)*

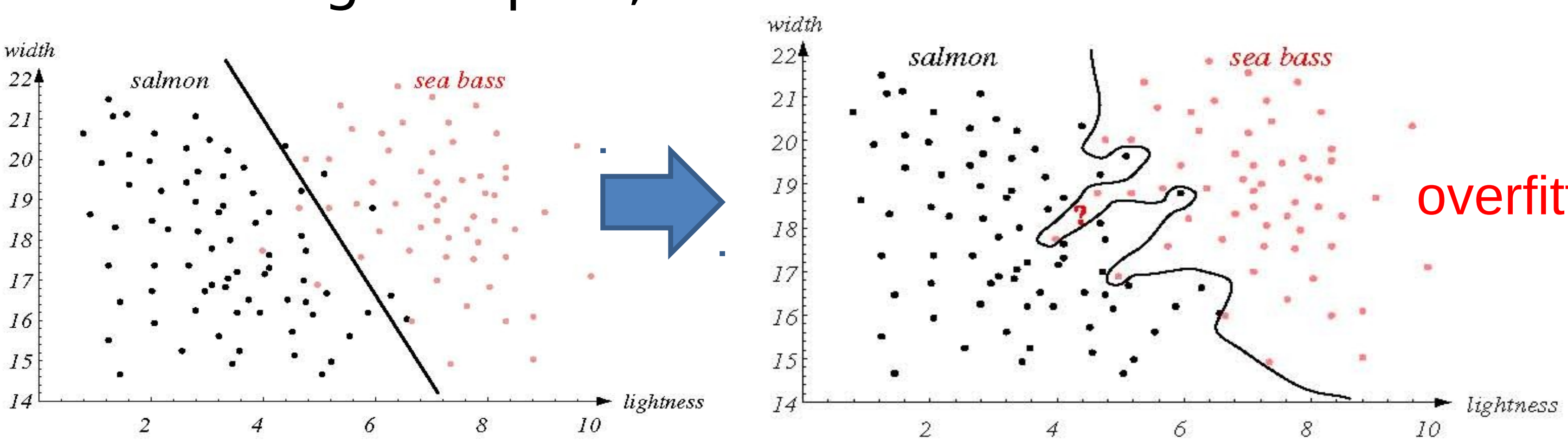
y: *class label (class)*

- *Generative*
  - Model the joint probability, p(x, y)
  - Make predictions by using Bayes rules to calculate p(ylx)
  - Pick the most likely label y
  - Eg: Gaussian, Naïve Bayes, Bayesian networks, HMM

- *Discriminative*
  - Estimate p(ylx) directly (e.g., learn a direct map from inputs x to the class labels y)
  - Pick the most likely label y
  - Eg: logistic regression, SVM, neural networks, nearest neighbor

Complexity

- We can get perfect classification performance on the training data by choosing complex models.
- Complex models are tuned to the particular training samples, rather than on the



overfitting

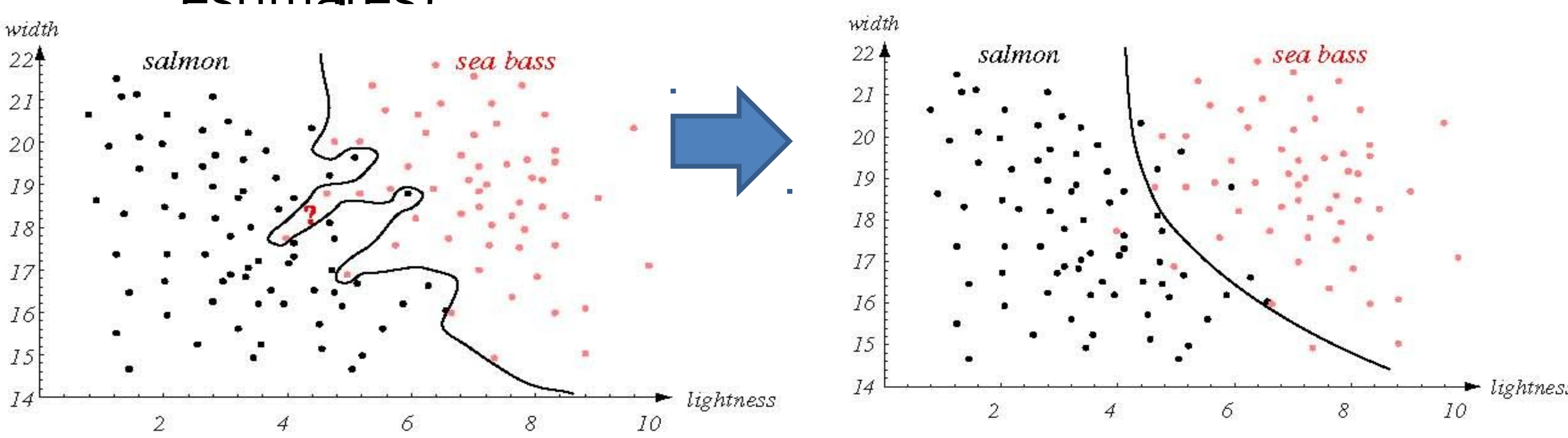
How well can the model generalize to unknown samples?

Generalization

- Generalization is defined as the ability of a classifier to produce correct results on novel patterns.
- How can we improve generalization performance ?
  - More training examples (i.e., better model estimates)

complex model

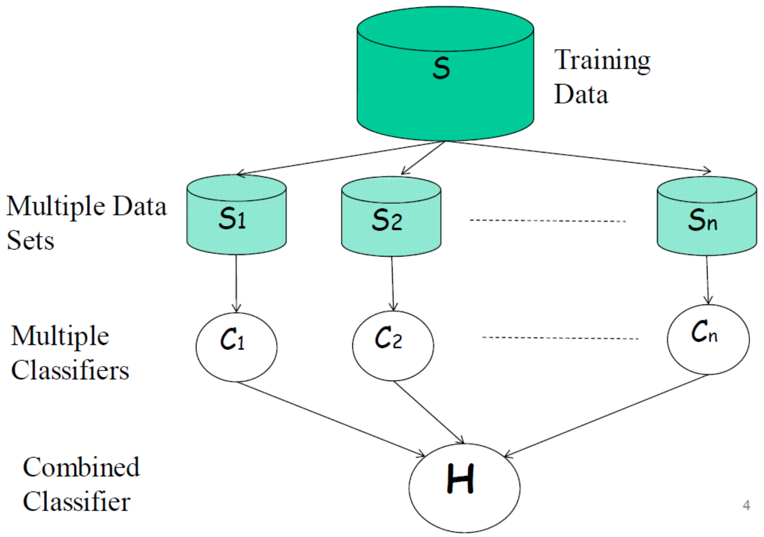
simpler model



Ensembles of Classifiers

- Performance can be improved using a "pool" of classifiers.

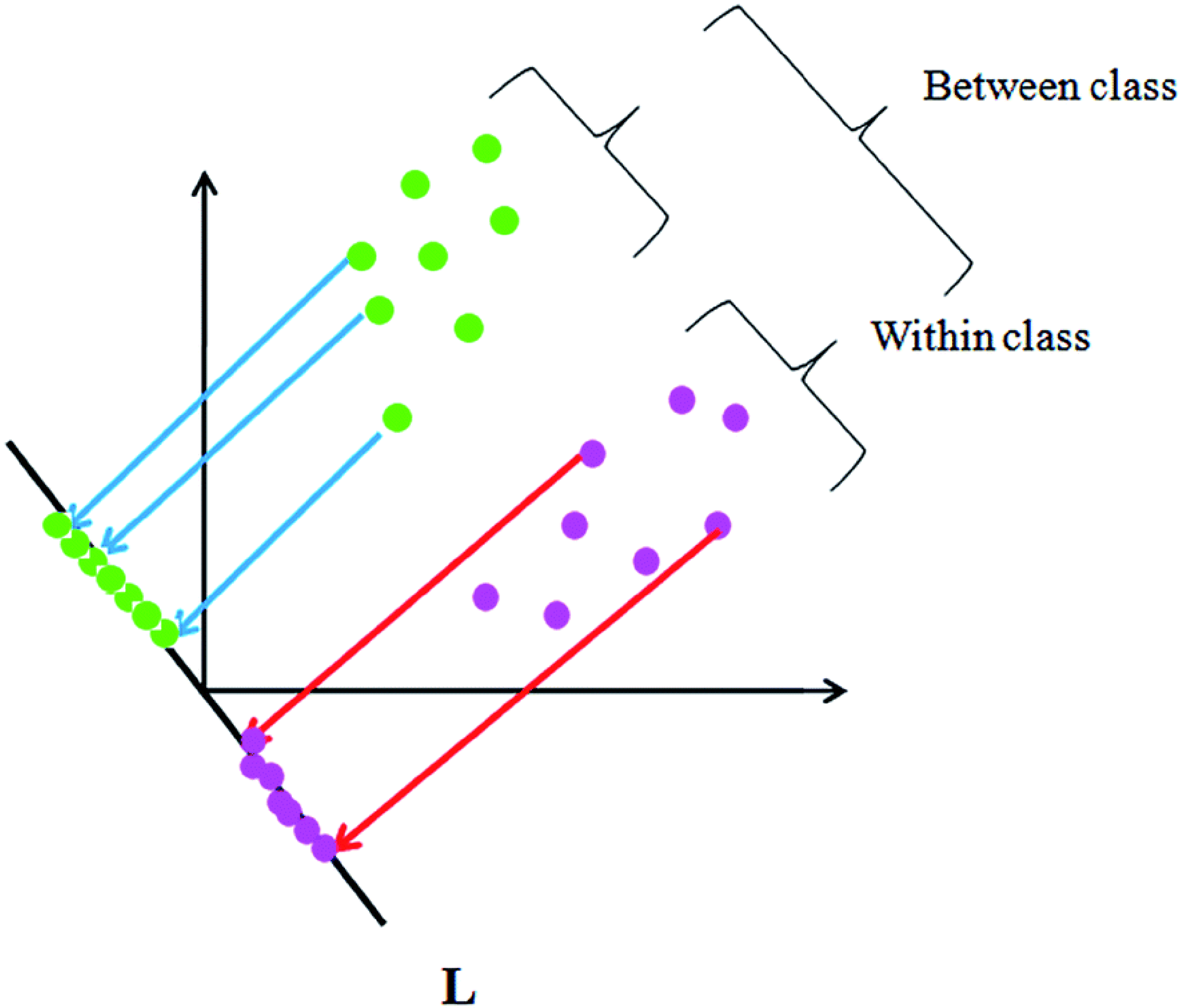
- How should we build and combine different classifiers ?

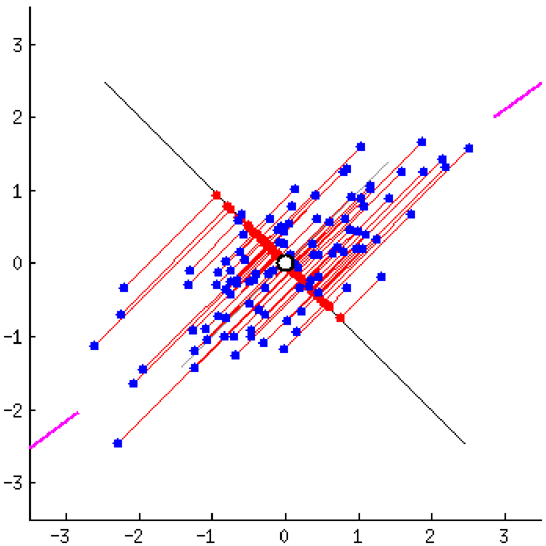


Would it be possible to build a “general purpose” PR system?

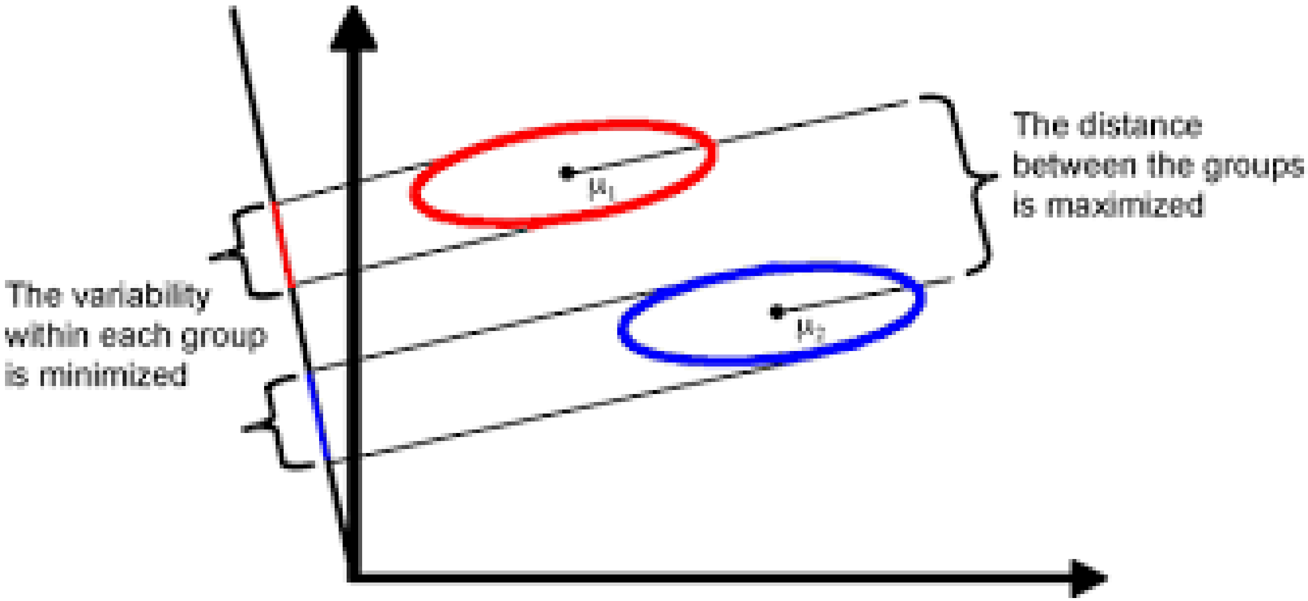
- No. Humans have the ability to switch rapidly and seamlessly between different pattern recognition tasks.
- It is very difficult to design a system that is capable of performing a variety of classification tasks.
  - Different decision tasks may require different features.
  - Different features might yield different solutions.
  - Different tradeoffs exist for different tasks.

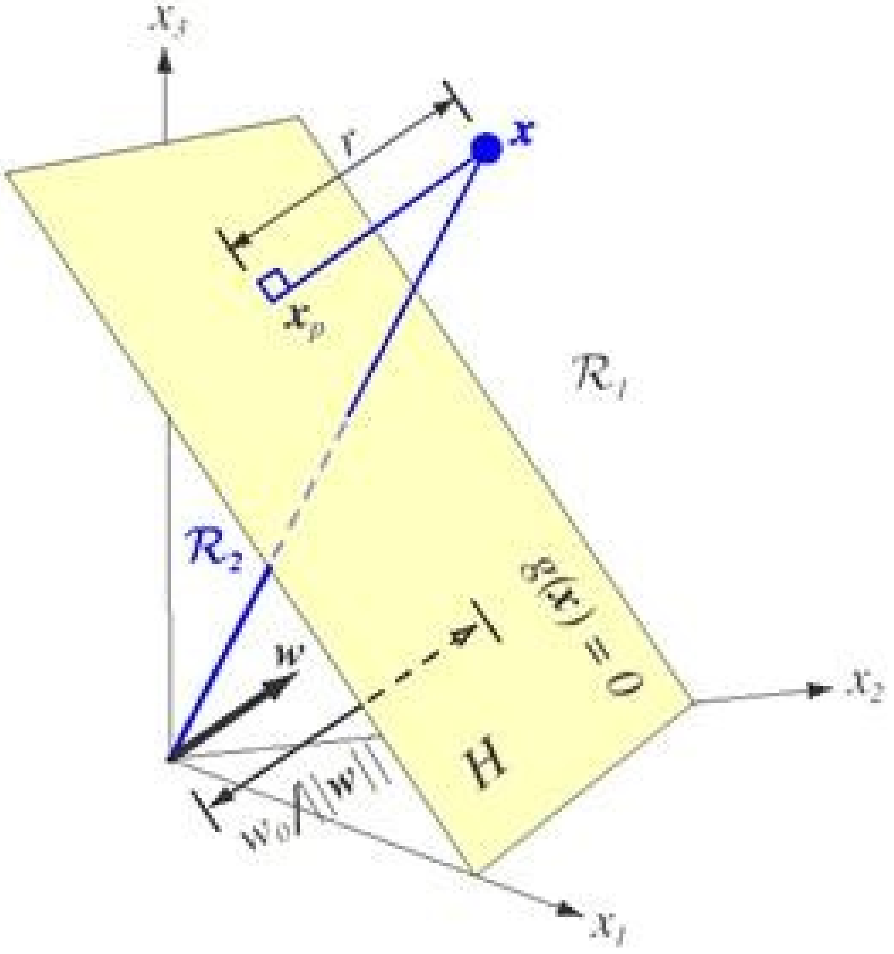
Thank you


Between class
Within class
L



3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


The distance
between the groups
is maximized
μ_1
μ_2
The variability
within each group
is minimized



x_3
r
$\boldsymbol{x}$
$\boldsymbol{x}_p$
$\mathcal{R}_1$
$\mathcal{R}_2$
$g(\boldsymbol{x}) = 0$
$\boldsymbol{w}$
H
$w_0/||\boldsymbol{w}||$
x_2
x_1