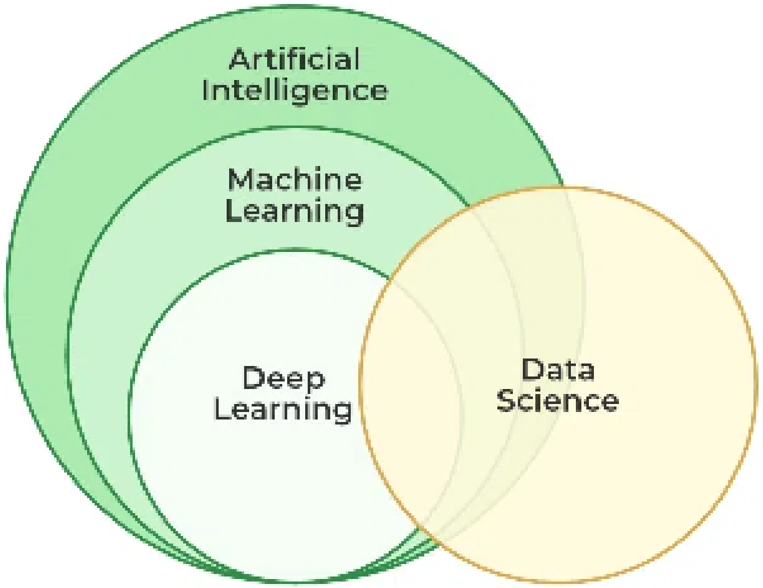
Introduction to Machine Learning

Badri Narayan Subudhi
IIT Jammu

Machine Learning



- Machine learning is a branch of AI focused on building computer systems that learn from data. The breadth of ML techniques enables software applications to improve their performance over time.
- Traditional machine learning combines data with statistical tools to predict outputs, yielding actionable insights.

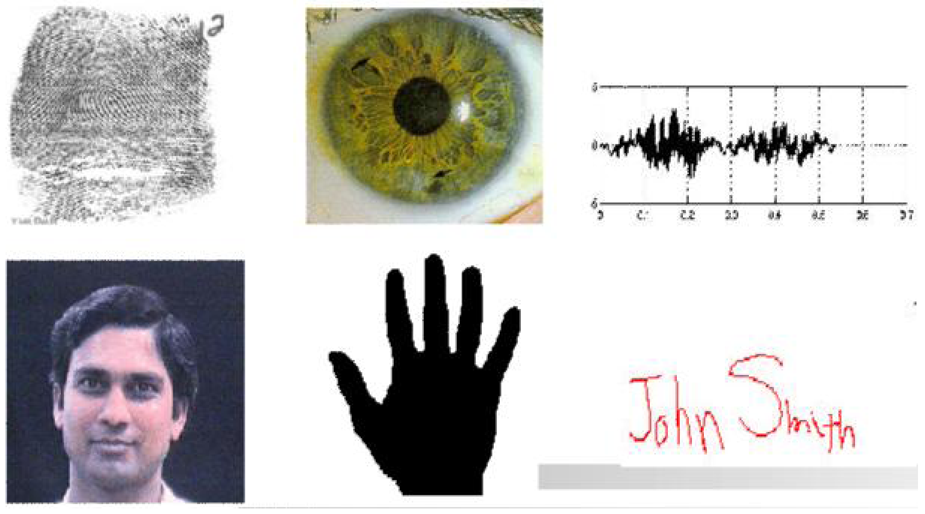
What is Machine Learning?

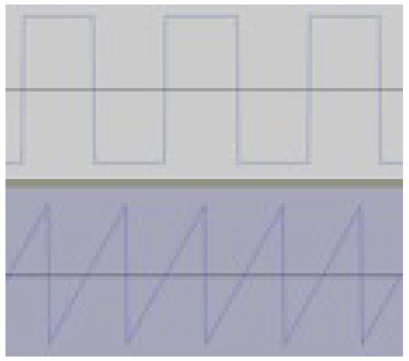
- Pattern Recognition is the study of how machines can:
  - observe the environment,
  - learn to distinguish patterns of interest,
  - make sound and reasonable decisions about the categories of the patterns.

- What is a *pattern*?
- What kinds of *category* we have?

What is a pattern?

- A **pattern** is an abstraction, represented by a set of measurements describing a “physical” object.
- Many types of patterns exist:
  - visual, temporal, sonic, logical, ...

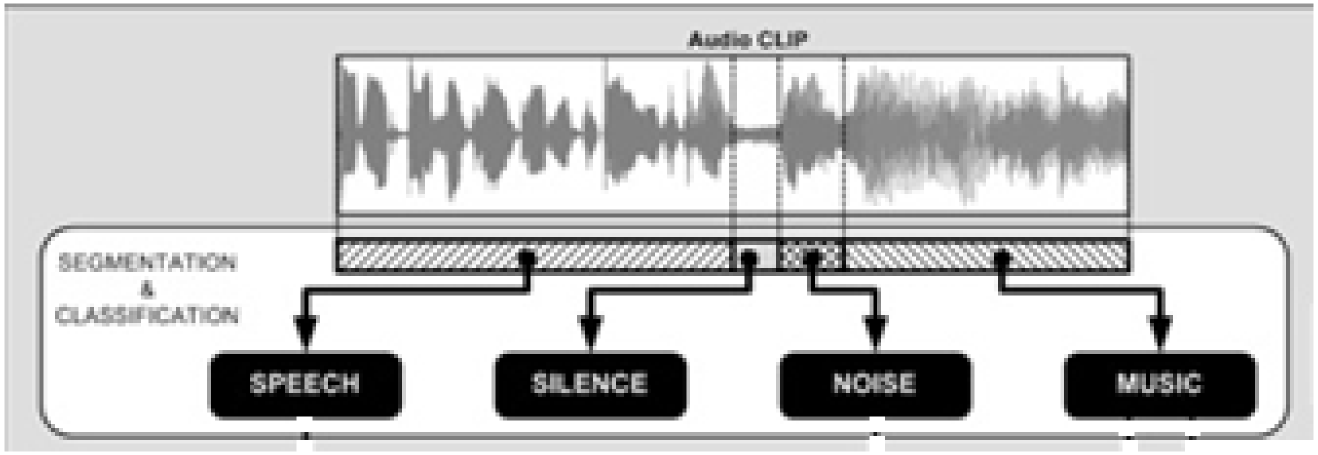




What is a Pattern Class (or category)?

– Is a set of patterns sharing common attributes

– A collection of “similar”, not necessarily identical, objects

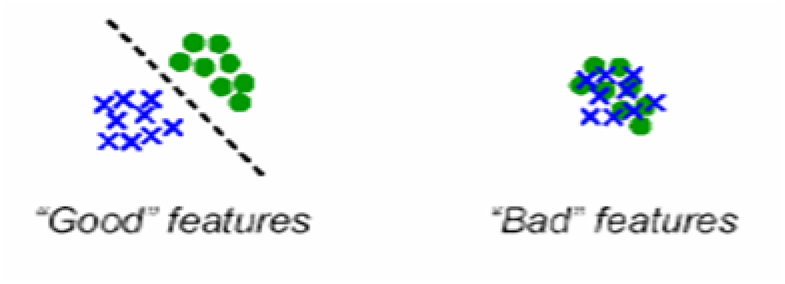


What is **Features** ?

- Features are properties of an object:
  - Ideally representative of a specific type (i.e. class) of objects
  - Perceptual relevant

Good features vs. bad features

- Ideally, for a given group of patterns coming from the same class, feature values should all be similar
- For patterns coming from different classes, the feature values should be different.



Aapplications

Recognition (OCR) → Printed texts: reading machines for blind people, digitalization of text documents.

Biometrics →
- Face recognition, verification, retrieval.
- Finger prints recognition.
- Speech recognition.

Diagnostic systems →
- Medical diagnosis: X-Ray, ECG analysis.
- Machine diagnostics, waster detection.

Target Tracking → Object detection and tracking from videos

Classification in Statistical ML

- A feature extractor is a program that inputs the data (image) and extracts features that can be used in classification.

- A classifier is a program that inputs the feature vector and assigns it to one of a set of designated classes or to the “reject” class.

With what kinds of classes do you work?

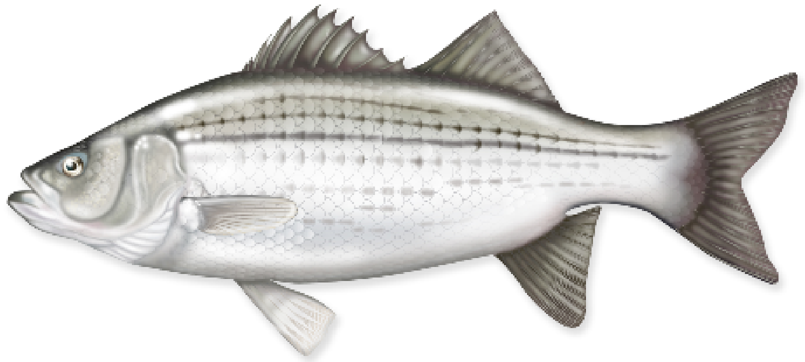
ML Process

- **Data acquisition and sensing:**
  - Measurements of physical variables.
  - Important issues: bandwidth, resolution , etc.
- **Pre-processing:**
  - Feature selection.
  - Isolation of patterns of interest from the background.
- **Feature extraction:**
  - Finding a new representation in terms of features.
- **Classification**
  - Using features and learned models to assign a pattern to a category.

An Example

“Sorting incoming Fish on a conveyor according to species using optical sensing”

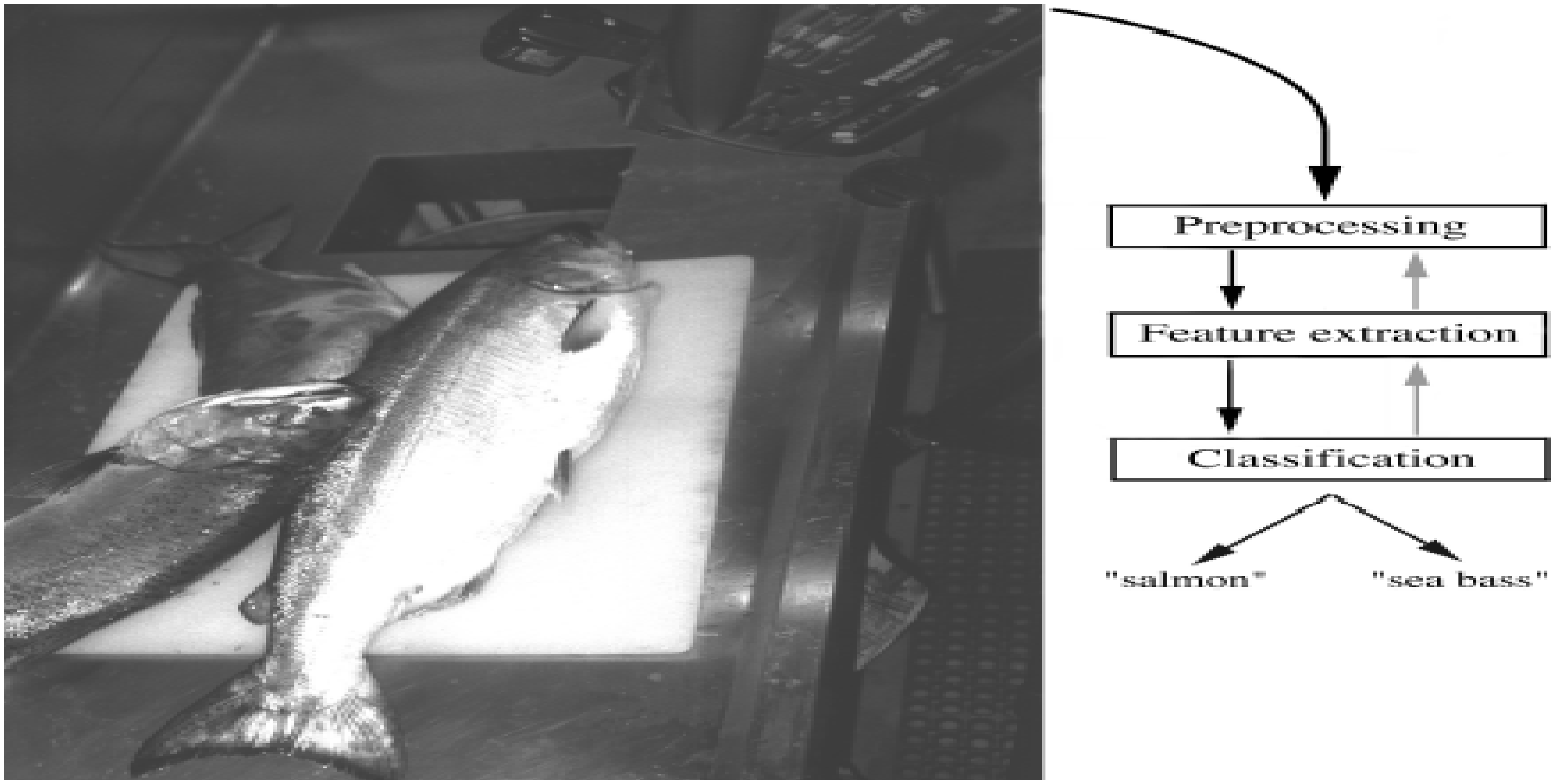
$$\begin{tikzcd}
 & \text{Sea bass} \\
\text{Species} \arrow[ru, dash] \arrow[rd, dash] & \\
 & \text{Salmon}
\end{tikzcd}$$



- Problem Analysis
  - Set up a camera and take some sample images to extract features
    - Length
    - Lightness
    - Width
    - Number and shape of fins
    - Position of the mouth, etc…

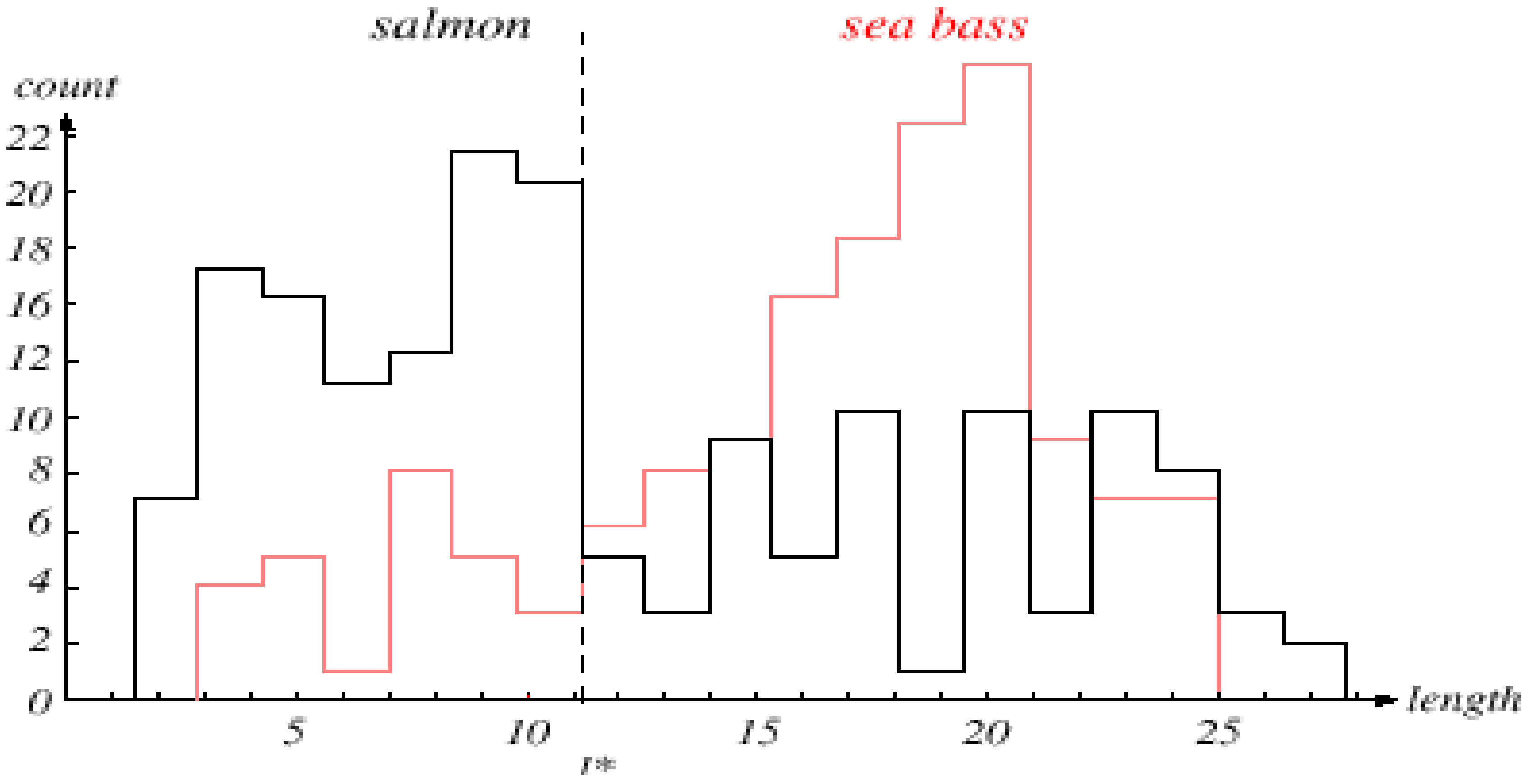
This is the set of all suggested features to explore for use in our classifier!

- Preprocessing
  - Use a segmentation operation to isolate fishes from one another and from the background
- Information from a single fish is sent to a feature extractor whose purpose is to represent the data by measuring certain features
- The features are passed to a classifier


Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

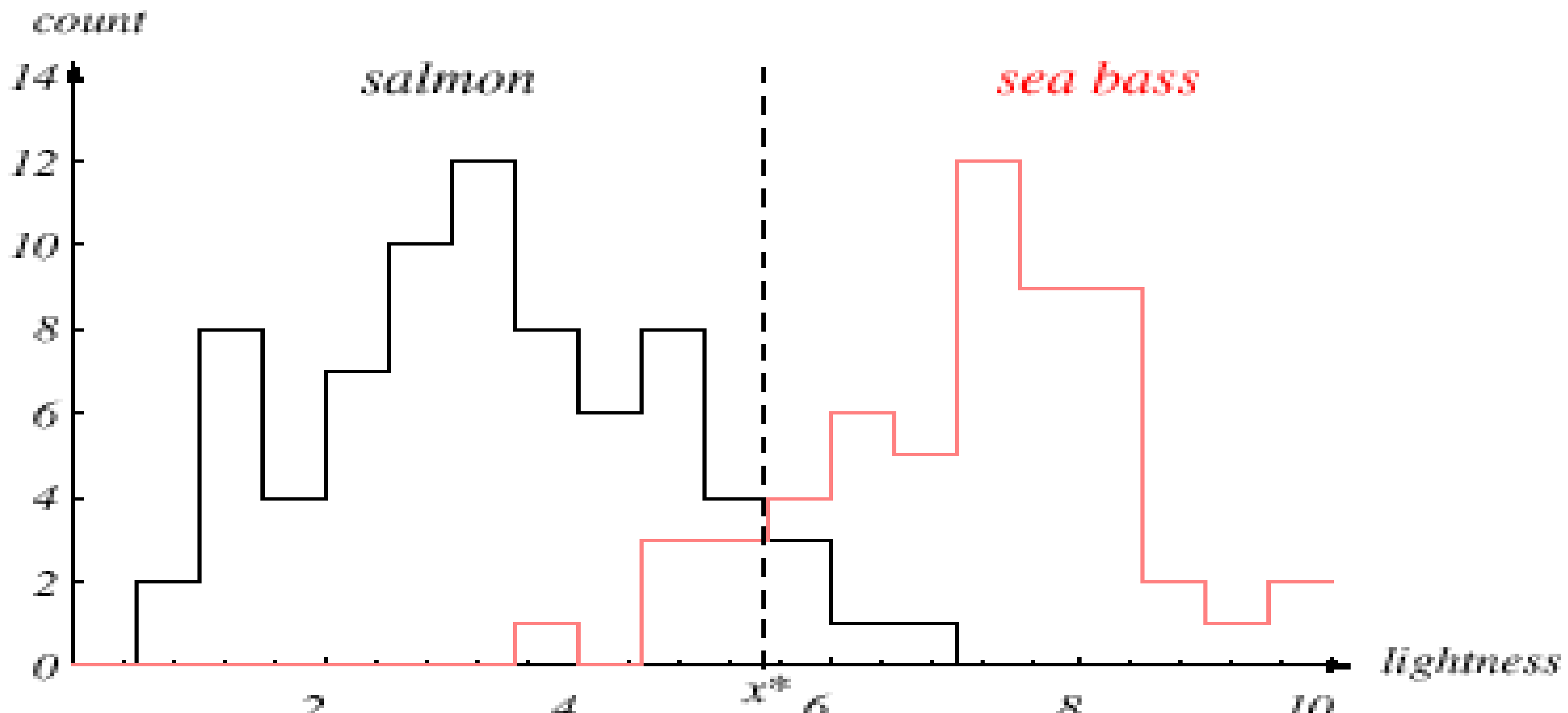
- Classification

Select the length of the fish as a possible feature for discrimination


salmon
sea bass
count
22
20
18
16
12
10
8
6
4
2
0
5
10
15
20
25
length
l*

The length is a poor feature alone!

Select the lightness as a possible feature.

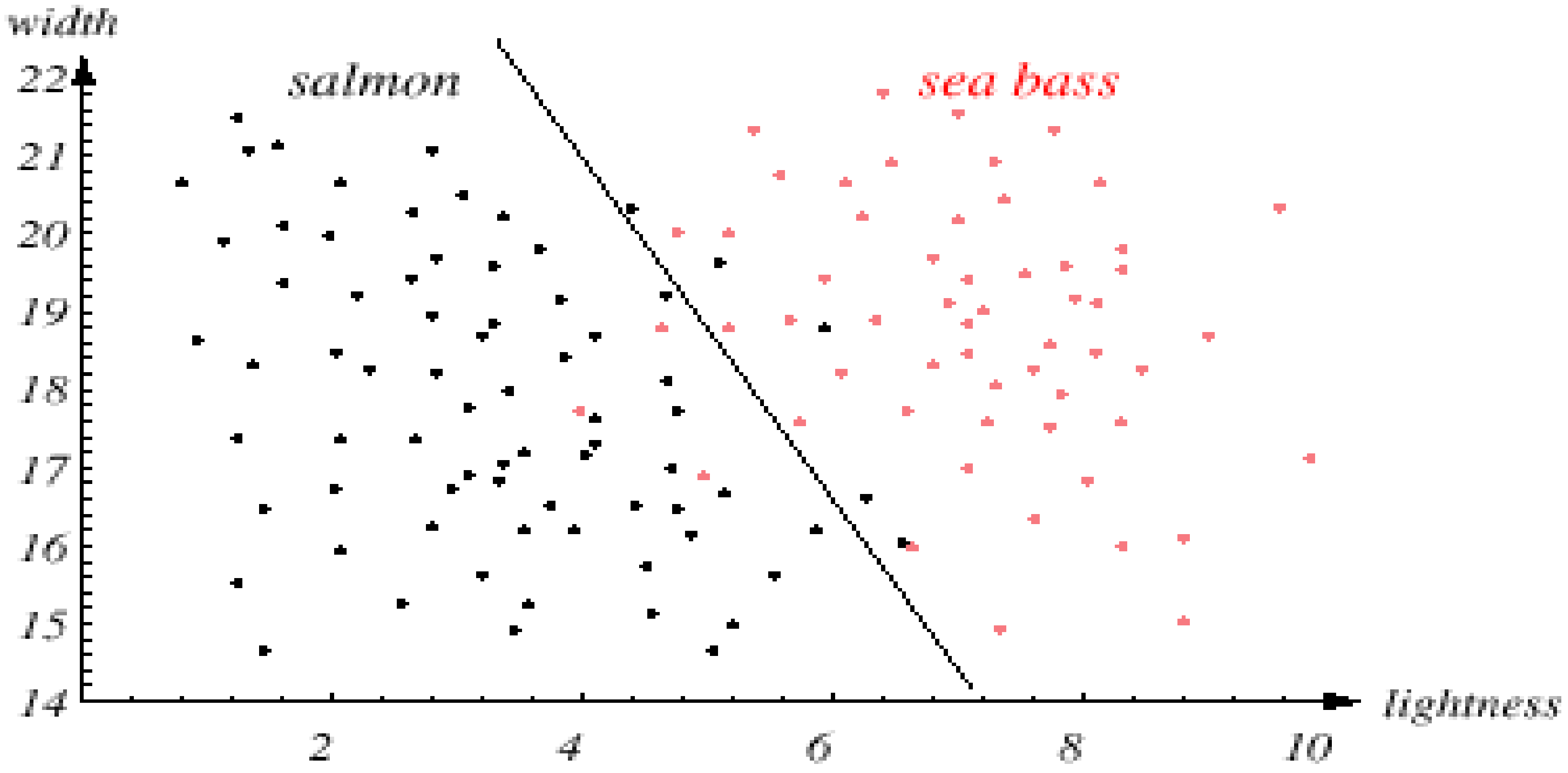

count
14
12
10
8
6
4
2
0
salmon
sea bass
lightness
2
4
x*
6
8
10

- Threshold decision boundary and cost relationship
  - Move our decision boundary toward smaller values of lightness in order to minimize the cost (reduce the number of sea bass that are classified salmon!)

Task of decision theory

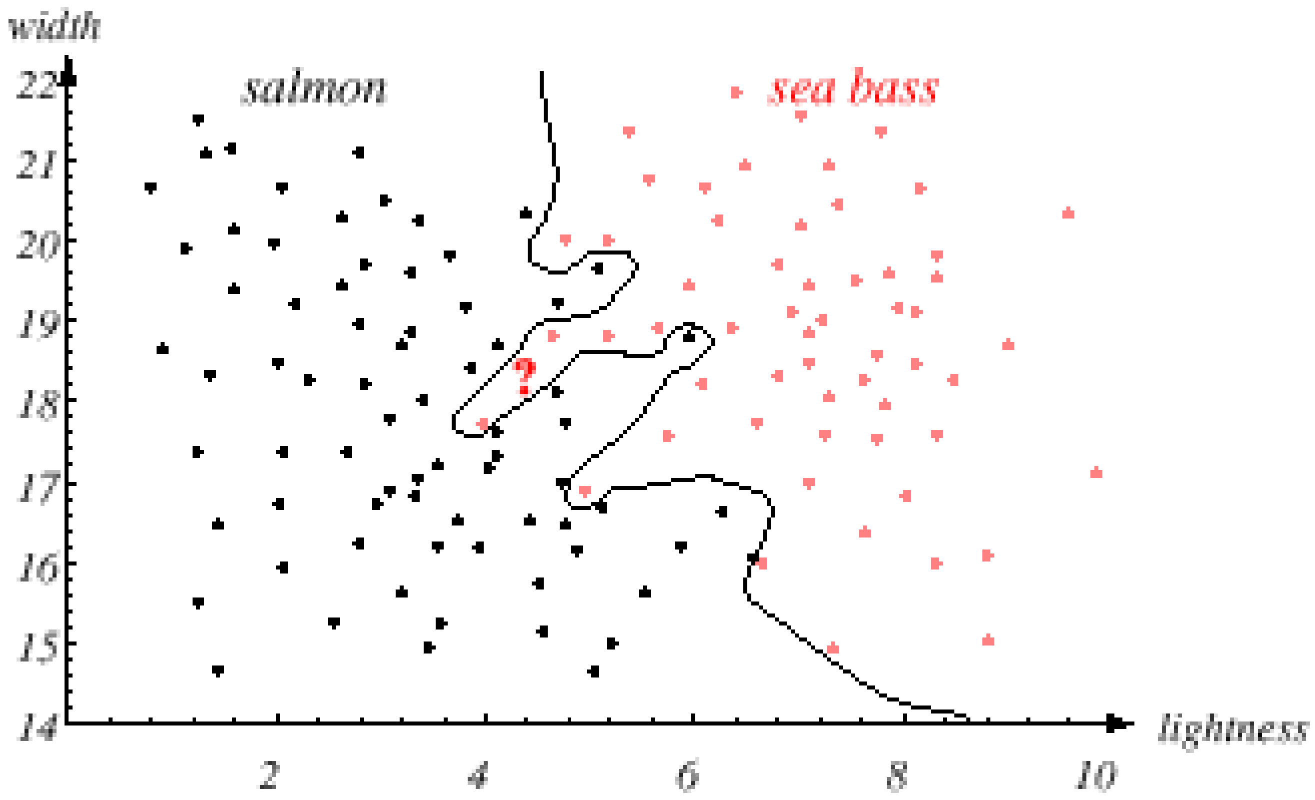
- Adopt the lightness and add the width of the fish

$$\begin{tikzcd}
\text{Fish} \arrow[r] & x = [x_1, x_2] \arrow[d] \arrow[dr] & \\
 & \textbf{Lightness} & \textbf{Width}
\end{tikzcd}$$


width
22
21
20
19
18
17
16
15
14
salmon
sea bass
lightness
2
4
6
8
10

- We might add other features that are not correlated with the ones we already have. A precaution should be taken not to reduce the performance by adding such “noisy features”

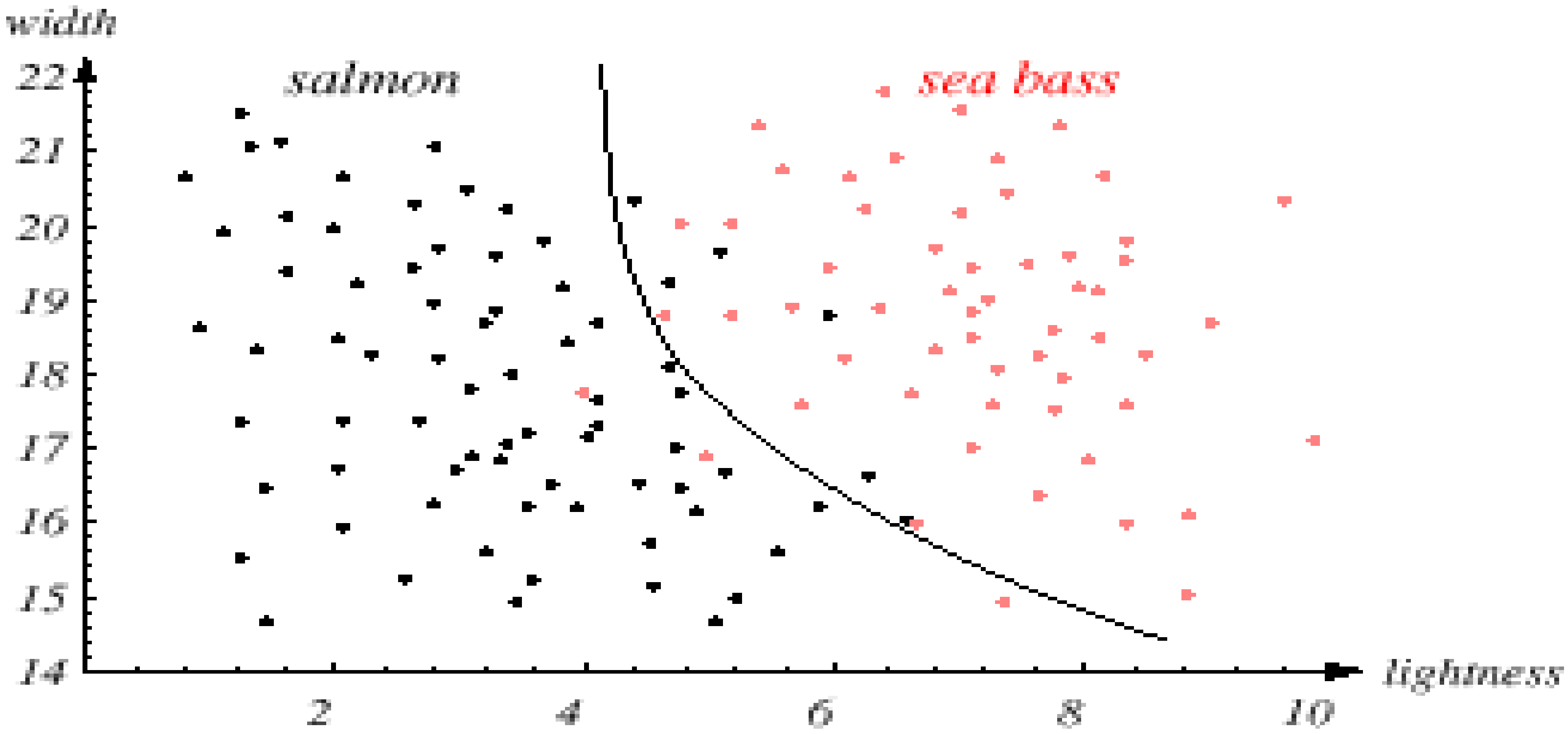
- Ideally, the best decision boundary should be the one which provides an optimal performance such as in the following figure:



width
salmon
sea bass
?
22
21
20
19
18
17
16
15
14
2
4
6
8
10
lightness

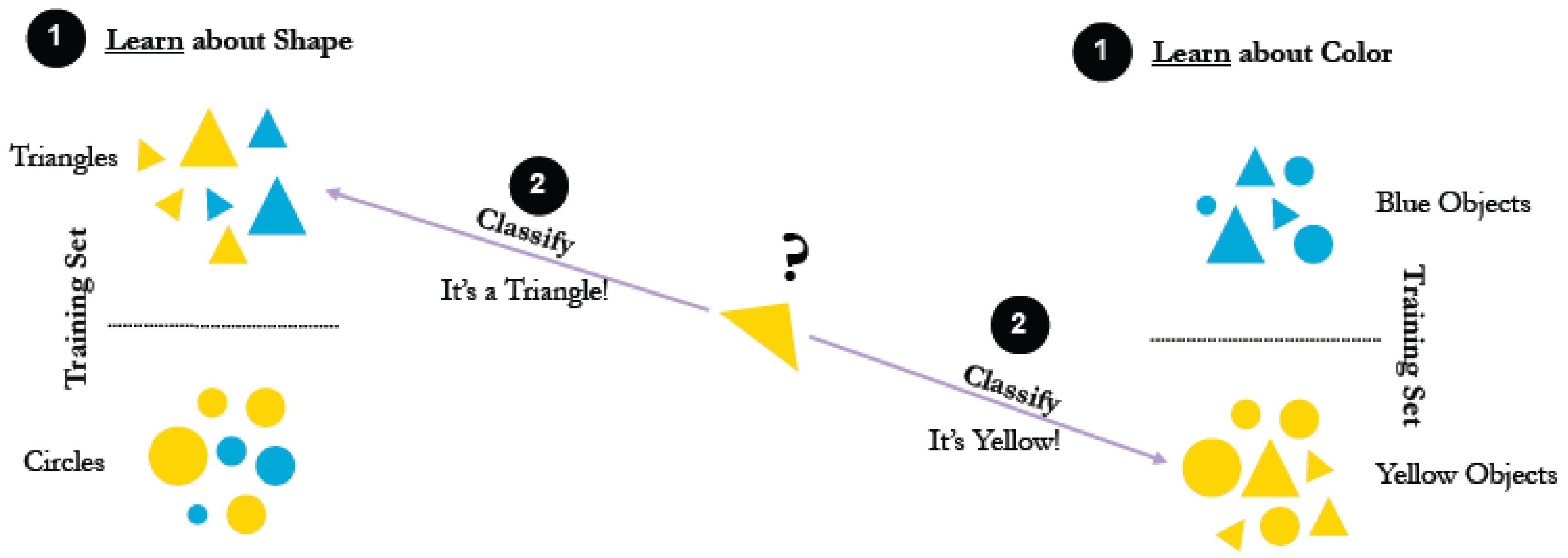
- However, our satisfaction is premature because the central aim of designing a classifier is to correctly classify novel input

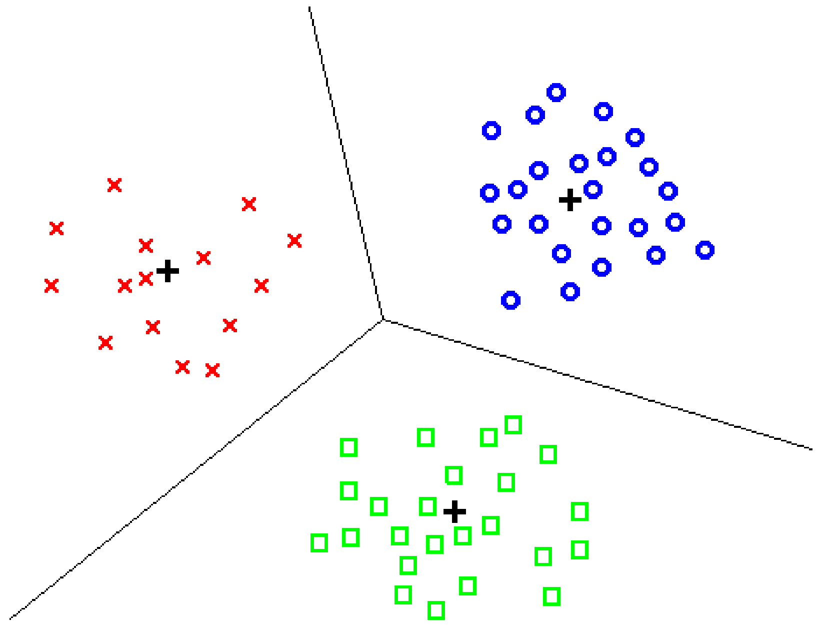
Issue of generalization!


width
salmon
sea bass
22
21
20
19
18
17
16
15
14
2
4
6
8
10
lightness

Supervised Training/Learning

– a "teacher" provides labeled training sets, used to train a classifier

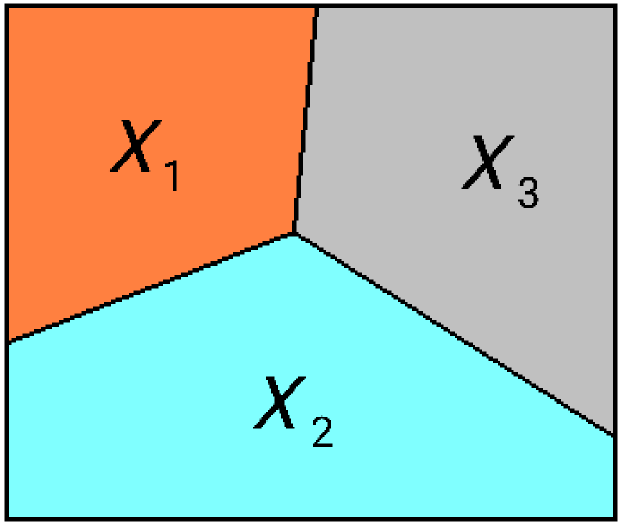


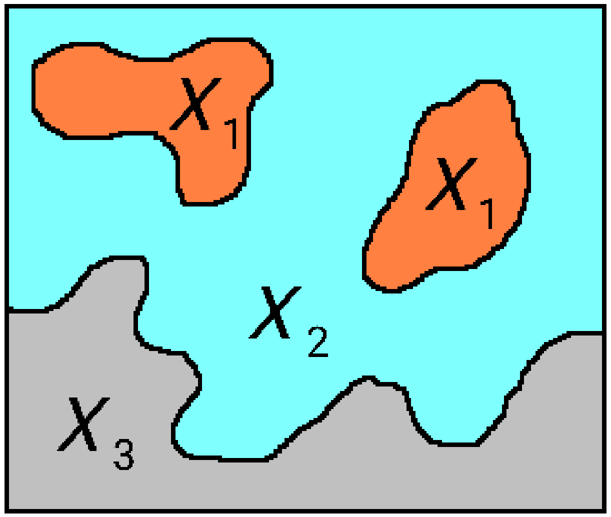


Classifier

A classifier partitions sample space X into **class-labeled regions** such that

$X = X_1 \cup X_2 \cup \ldots \cup X_{|Y|}$ and $X_i \cap X_j = \{\varphi\}$

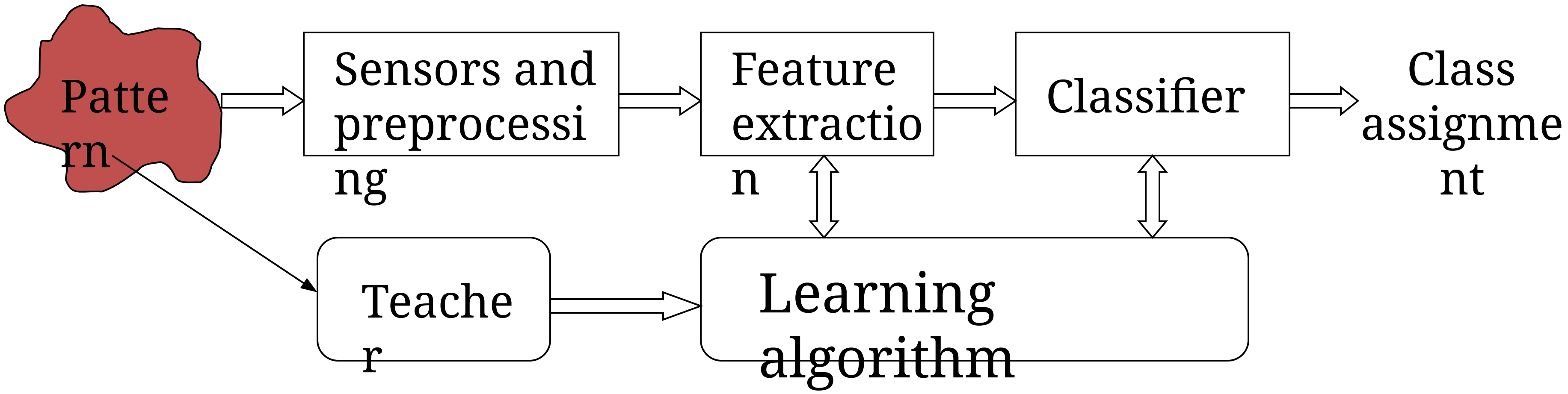




The classification consists of determining to which region a feature vector **x** belongs to.

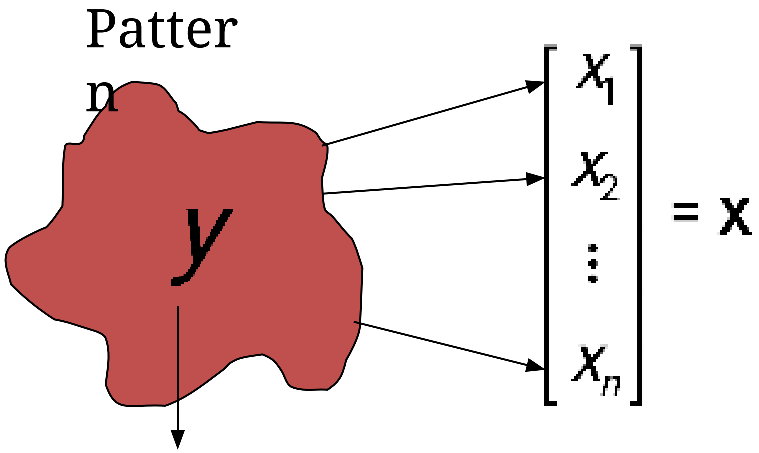
Borders between **decision boundaries** are called decision regions.

Components of classifier system



- **Sensors and preprocessing**.
- **A feature extraction** aims to create discriminative features good for classification.
- **A classifier**.
- **A teacher** provides information about hidden state -- supervised learning.
- **A learning algorithm** sets PR from training examples.

Basic concepts



Pattern
y
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \mathbf{x}$$

Feature vector $\mathbf{x} \in X$

- A vector of observations (measurements).

- $\mathbf{x}$ is a point in feature space X .

Hidden state $y \in Y$

- Cannot be directly measured.

- Patterns with equal hidden state belong to the same class.

Task

- To design a classifer (decision rule) $q : X \rightarrow Y$

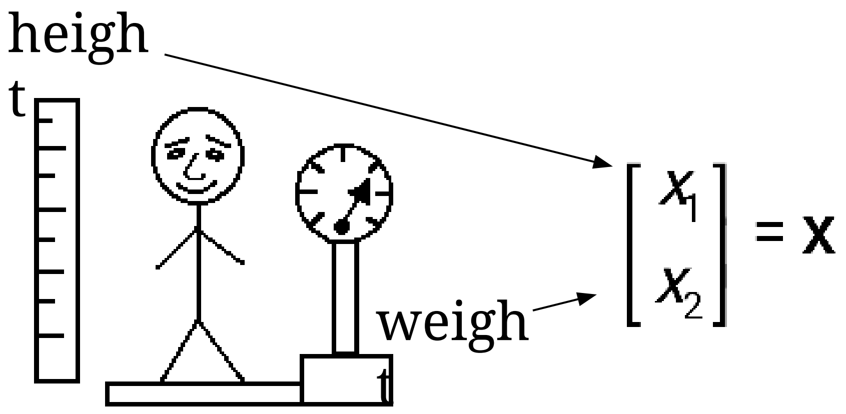
which decides about a hidden state based on an onbservation.

Training Samples

When a few labeled patterns can be collected by experts, Supervised learning can be opted instead of the unsupervised approach to make full utilization of labeled patterns.

In a dataset a training set can be implemented to build up a model.

Example



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

Task: American-Indian recognition.

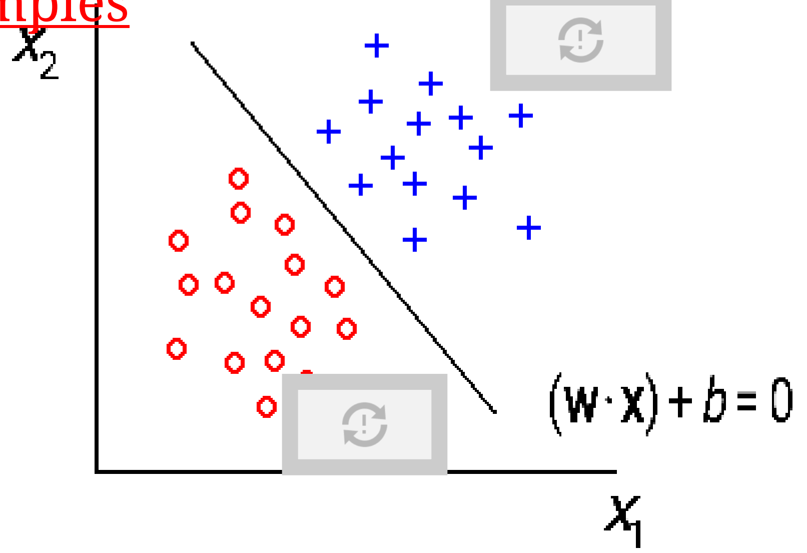
The set of hidden states is $\mathcal{Y} = \{A, I\}$

The feature space is $\mathcal{X} = \Re^2$

Training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$

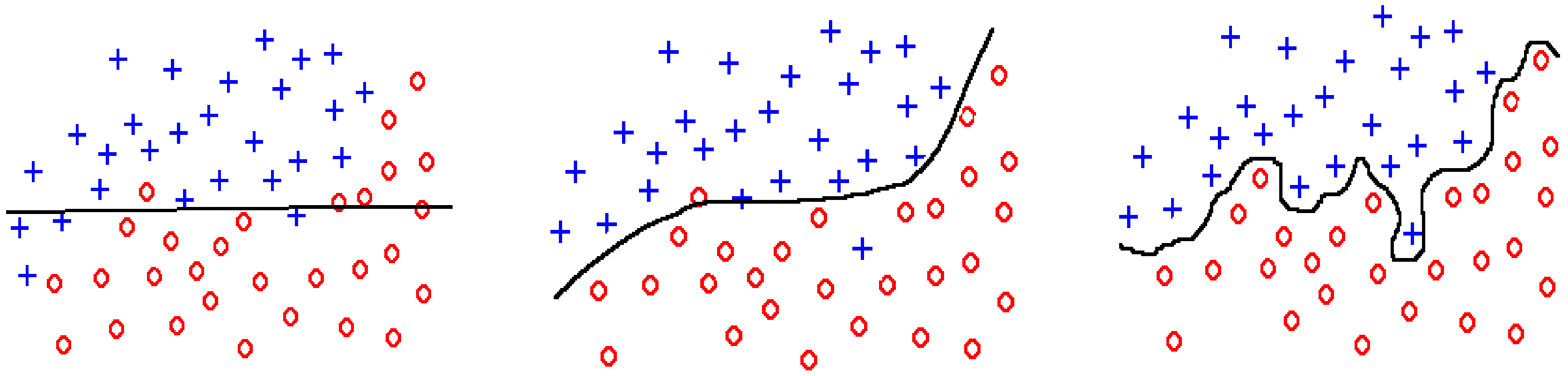
Linear classifier:

$$q(\mathbf{x}) = \begin{cases} A & if \quad (\mathbf{w} \cdot \mathbf{x}) + b \geq 0 \\ I & if \quad (\mathbf{w} \cdot \mathbf{x}) + b < 0 \end{cases}$$



Overfitting and underfitting

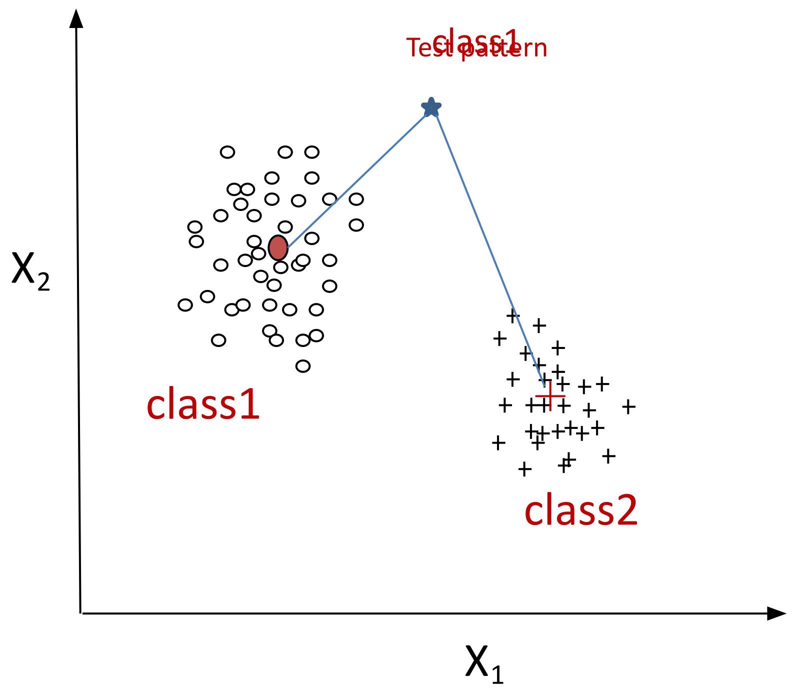
Problem: how rich class of classifications $q(\mathbf{x};\boldsymbol{\theta})$ to use.


underfitting good fit overfitting

Problem of generalization: a small emprical risk R_{emp} does not imply small true expected risk R.

Minimum Distance Classifier

- Each class is represented by its mean vector
- Training is performed by calculating the mean of the feature vectors of each class
- New patterns are classified by finding the closest mean vector
- The boundary is the perpendicular bisector of the line joining the mean points.



Minimum Distance Classifier

$$m_j = \frac{1}{N_j} \sum_{X \in \omega_j} X \qquad j = 1, 2, \cdots, M$$

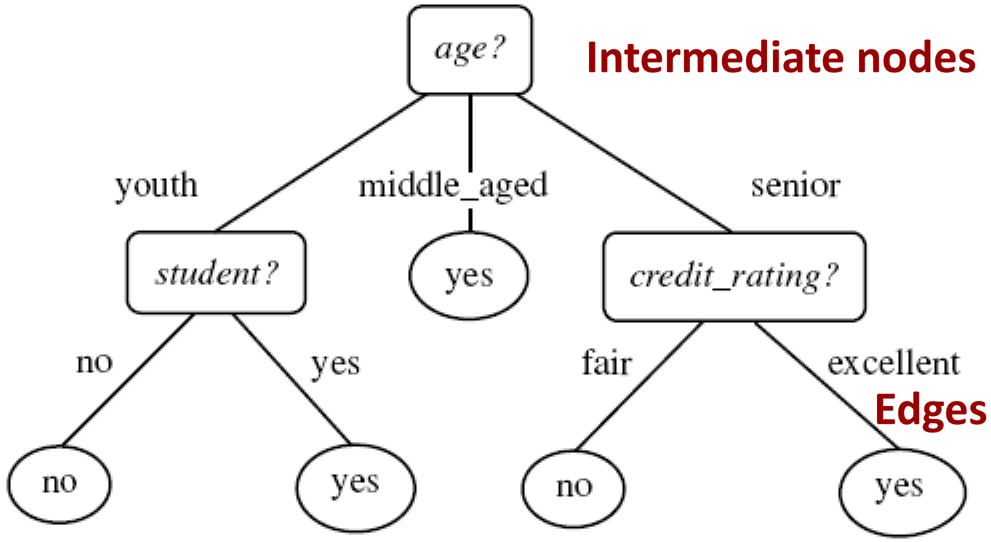
N_j = number of pixels from class ω_j

$$D_j(X) = \left\| X - m_j \right\| \qquad \text{(Euclidean distance)}$$

Assign X to ω_i *if* $D_i(X) < D_j(X), \quad j = 1, 2, \cdots, M;\ j \neq i$

Decision Trees

Example tree



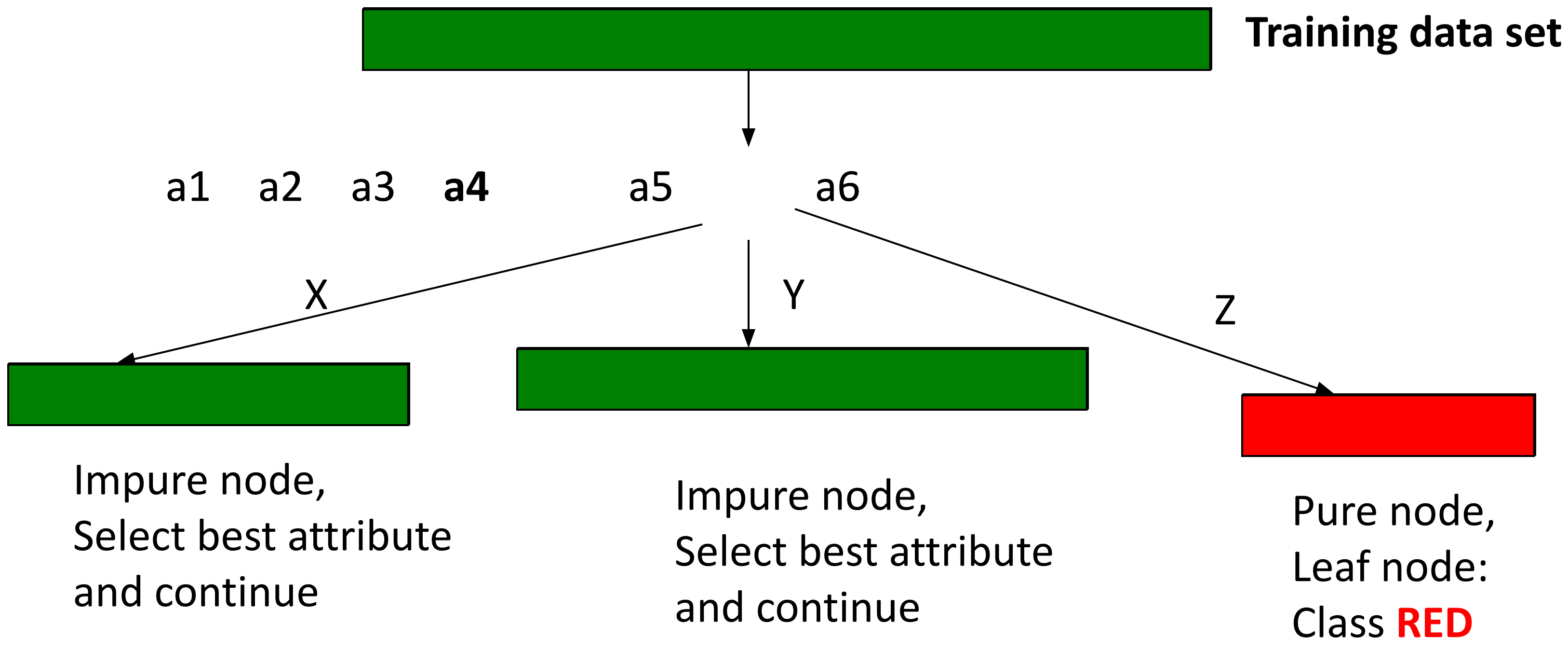
Intermediate nodes : Attributes

Edges : Attribute value tests

Leaf nodes : Class predictions

Example algorithms: ID3, C4.5, SPRINT, CART

Decision Tree schematic



Decision Tree Issues

How to determine the attribute for split?

Alternatives:

1. Information Gain

Gain (A, S) = Entropy (S) – **Σ** ((Sj/S)*Entropy(Sj))

Other options:

Gain ratio, etc.

Lazy learners

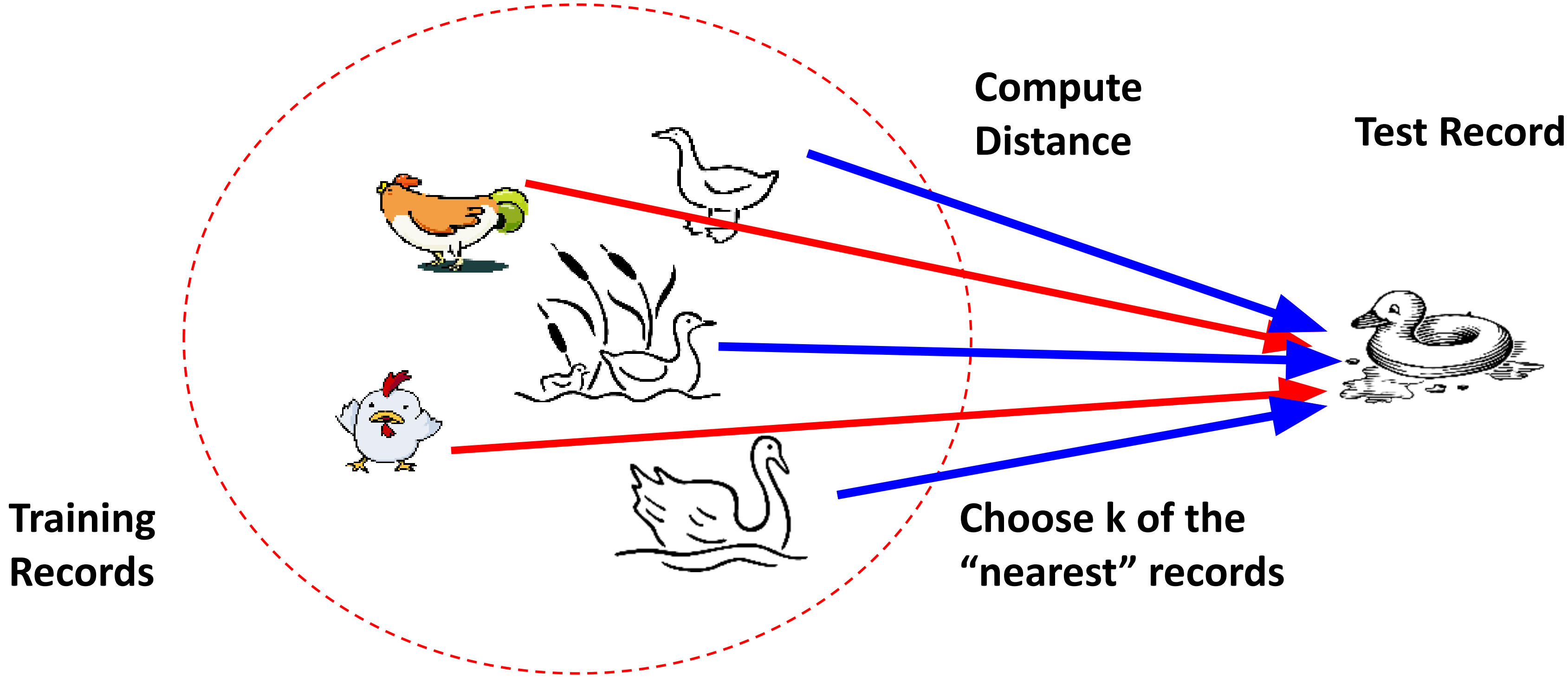
Lazy learners

- '**Lazy**': Do not create a model of the training instances in advance
- When an instance arrives for testing, runs the algorithm to get the class prediction
- Example, K – nearest neighbour classifier (K – NN classifier)

"One is known by the company one keeps"

Pattern-Based Classification: Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



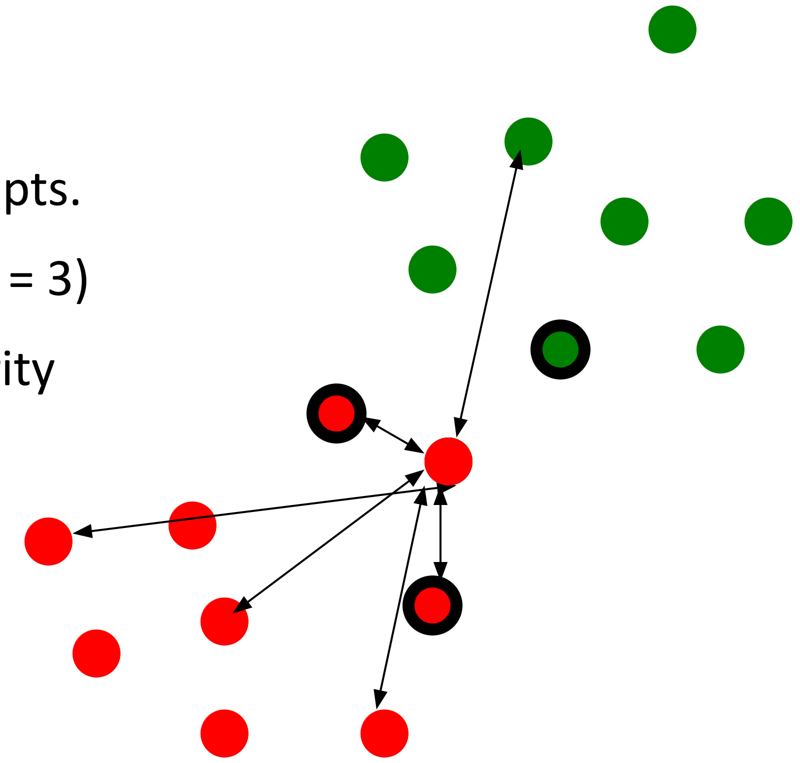
K-NN classifier schematic

For a test instance,

1) Calculate distances from training pts.
2) Find K-nearest neighbours (say, K = 3)
3) Assign class label based on majority

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

$$v' = \frac{v - min_A}{max_A - min_A},$$



K-NN classifier Issues

How to determine distances between values of categorical attributes?

Alternatives:

1. Boolean distance (1 if same, 0 if different)

2. Differential grading (e.g. weather – ‘drizzling’ and ‘rainy’ are closer than ‘rainy’ and ‘sunny’)

Bayesian Classification

We will pose the classification problem in probabilistic terms

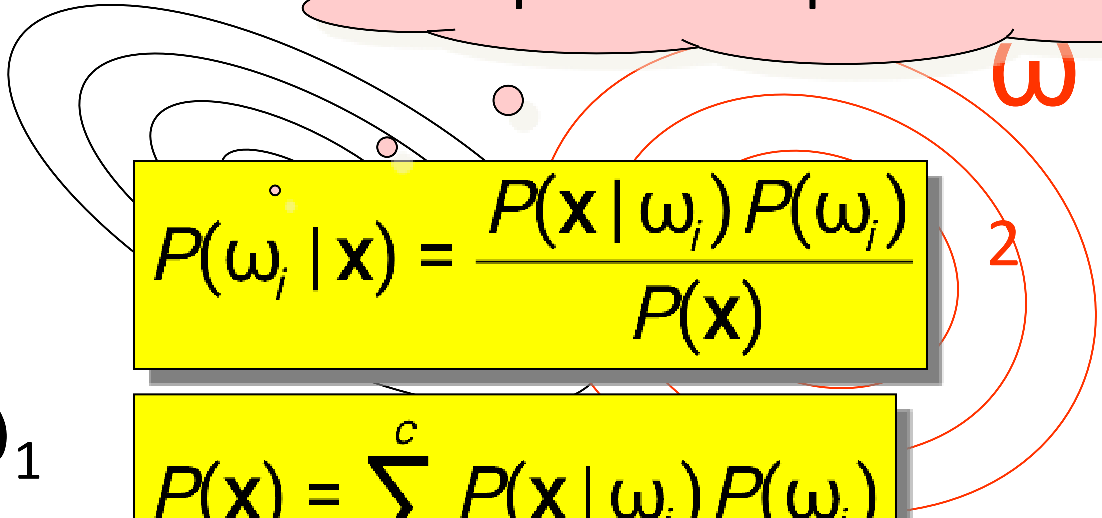
Create models for how features are distributed for objects of different classes

We will use probability calculus to make classification decisions

The States of Nature

$P(\mathbf{x}|\omega_i)$: class-conditional probab

A posterior probability

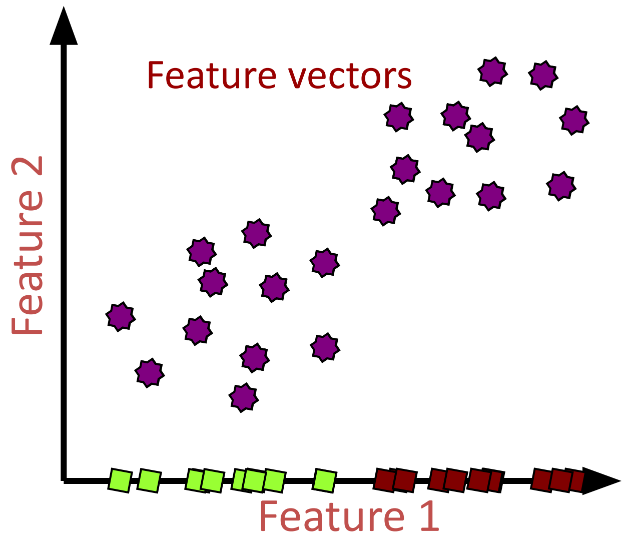


$$P(\omega_i|\mathbf{x}) = \frac{P(\mathbf{x}|\omega_i)P(\omega_i)}{P(\mathbf{x})}$$

$$P(\mathbf{x}) = \sum_{i=1}^{c} P(\mathbf{x}|\omega_i)P(\omega_i)$$

Lets Look at Just One Feature

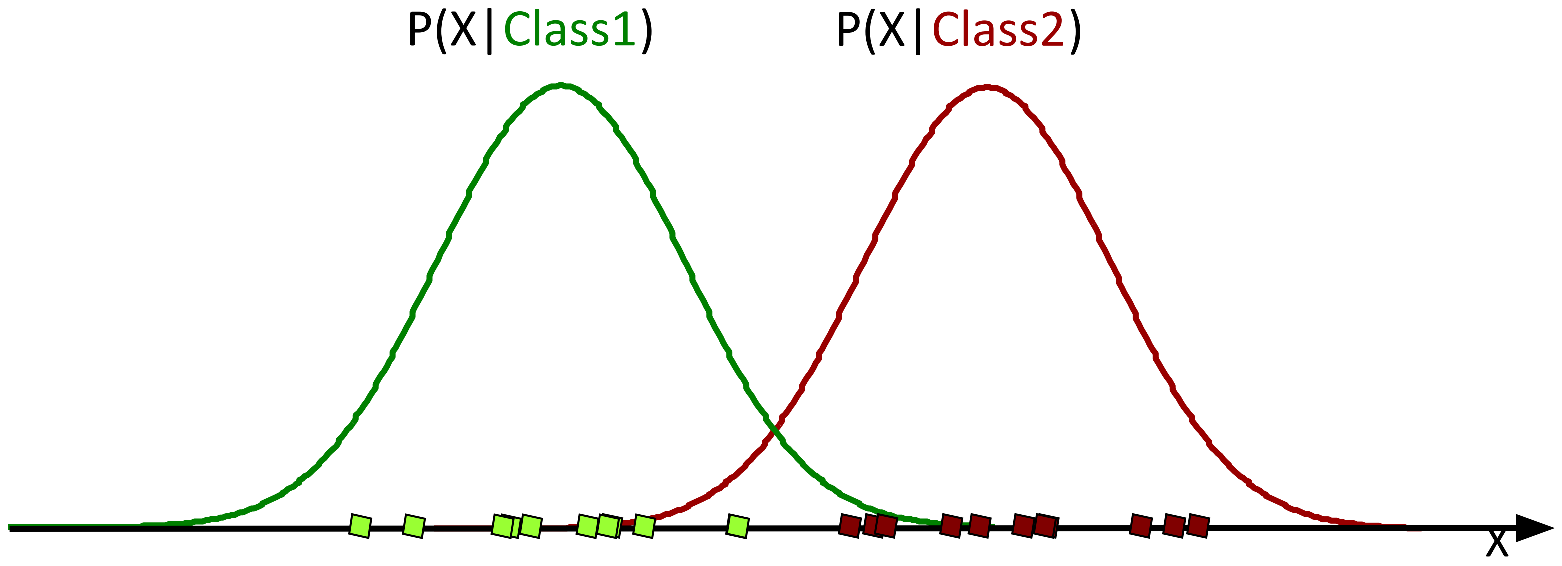
- Each object can be associated with multiple features
- We will look at the case of just one feature for now



We are going to define two key concepts….

The First Key Concept

Features for each class drawn from class-conditional probability distributions (CCPD)



Our first goal will be to *model* these distributions

The Second Key Concept

We model prior probabilities to quantify the expected *a priori* chance of seeing a class

$$P(\text{Class2}) \quad \& \quad P(\text{Class1})$$

But How Do We Classify?

- **So we have priors defining the *a priori* probability of a class**

P(Class1), P(Class2)

- **We also have models for the probability of a feature given each class**

P(X|Class1), P(X|Class2)

But we want the probability of the class given a feature
How do we get P(Class1|X)?

Bayes Rule

Likelihood

Evaluate evidence

Prior

Belief before evidence

$$P(Class \mid Feature) = \frac{P(Feature \mid Class)\,P(Class)}{P(Feature)}$$

Posterior

Belief after evidence

Evidence

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, 53:370-418

Bayes Decision Rule

If we observe an object with feature X, how do decide if the object is from Class 1?

The Bayes Decision Rule is simply choose Class1 if:

This is the same number on both sides!

Discriminant Function

We can create a convenient representation of the Bayes Decision Rule

$$P(X \mid Class1)\,P(Class1) > P(X \mid Class2)\,P(Class2)$$

If G(X) > 0, we classify as Class 1

Stepping back

What do we have so far?

We have defined the two components, class-conditional distributions and priors

$P(X|Class1), P(X|Class2)$ $P(Class1), P(Class2)$

We have used Bayes Rule to create a discriminant function for classification from these components

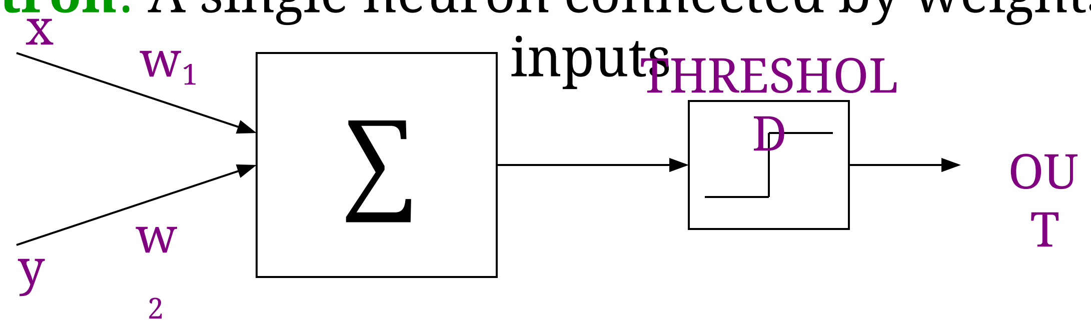
$$G(X) = \log \frac{P(X|Class1)}{P(X|Class2)} \frac{P(Class1)}{P(Class2)} > 0$$

Given a new feature, X, we plug it into this equation…

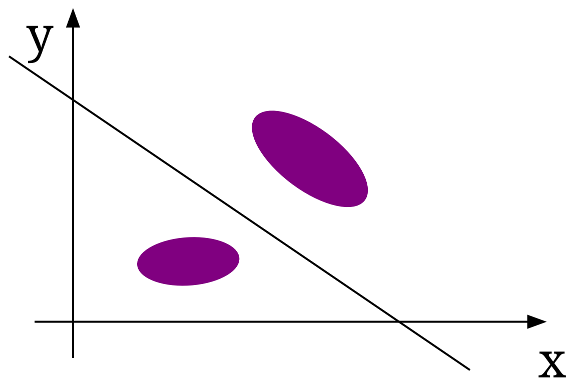
…and if G(X)> 0 we classify as Class1

Neural Classifier

Perceptron: A single neuron connected by weights to a set of inputs



- Let x & y be two inputs and w_1,w_2 be the weights.
- If $w_1x + w_2y > \theta$ then the output is **1** else **0**, where θ= threshold
    - $w_1x + w_2y = \theta \rightarrow$ separating line



Learning rule

Learning: Present a set of input patterns, adjust the weights until the desired output occurs for each of them.

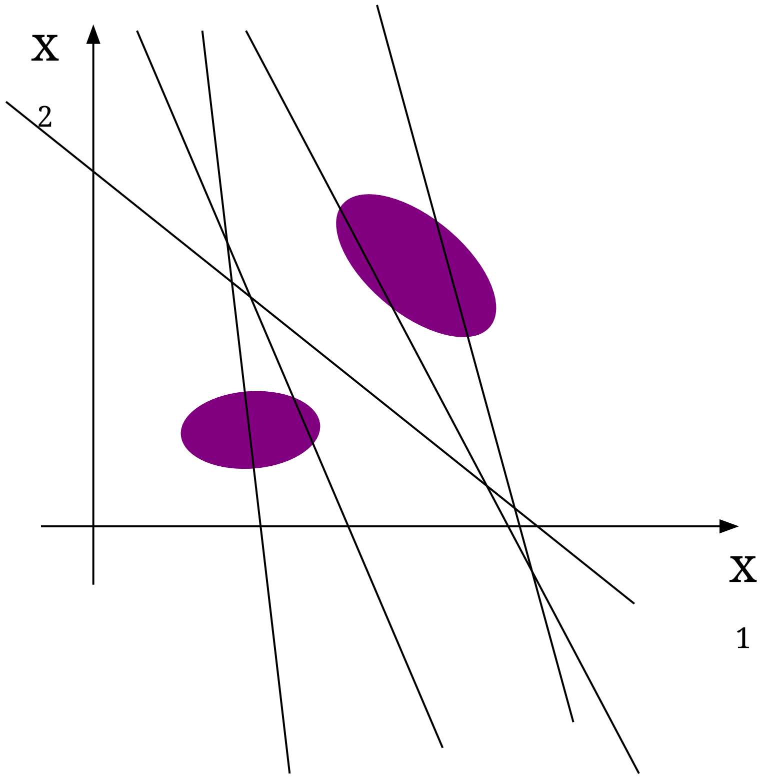
$$w_i(t+1) = w_i(t) + \Delta_i;$$

$$\Delta_i = \eta\ \delta\ x_i;$$

$$\delta \quad = T - A \text{ (i.e., target – actual).}$$

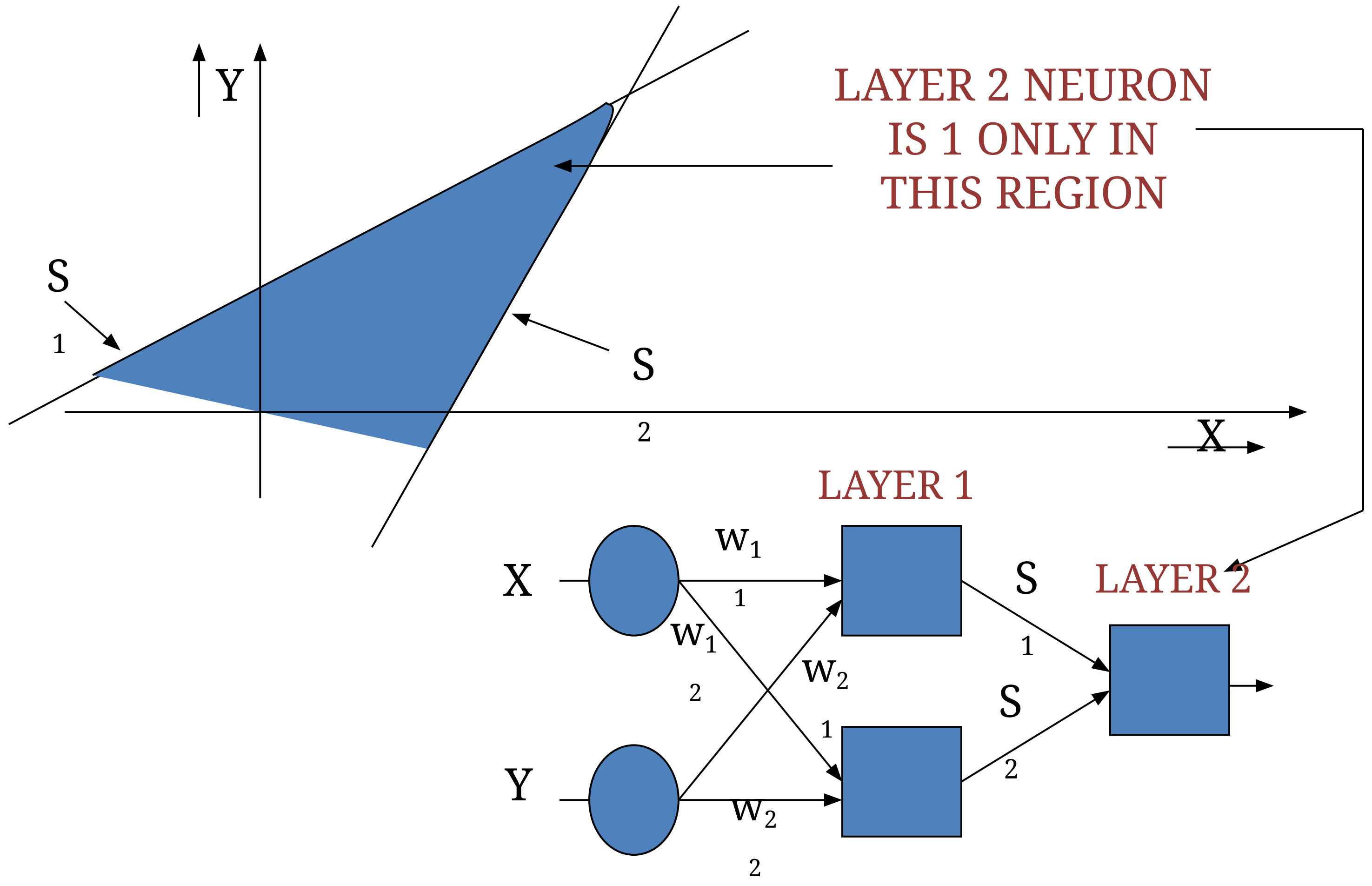
- If the sets of patterns are linearly separable, the single layer perceptron algorithm is guaranteed to find a separating hyperplane in a finite number of steps.

Change of weights

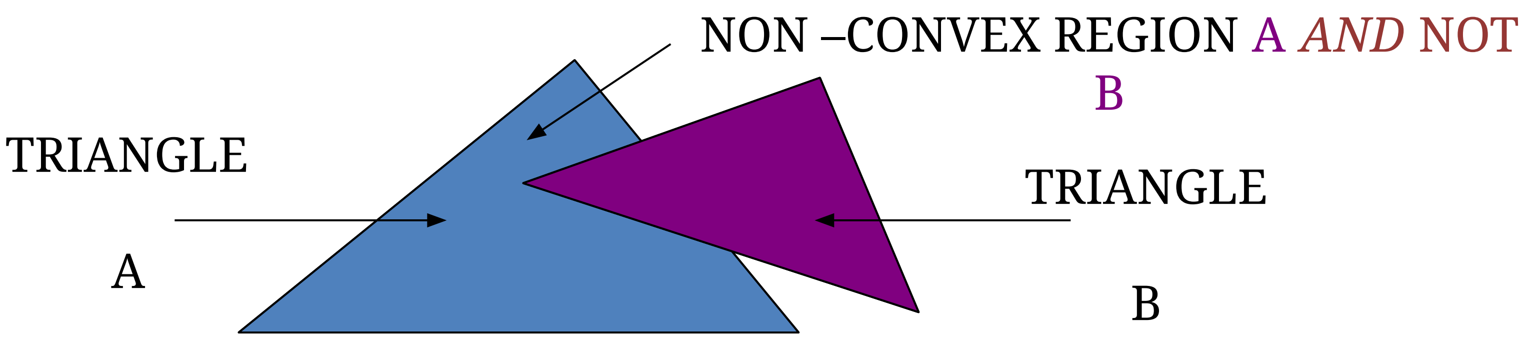


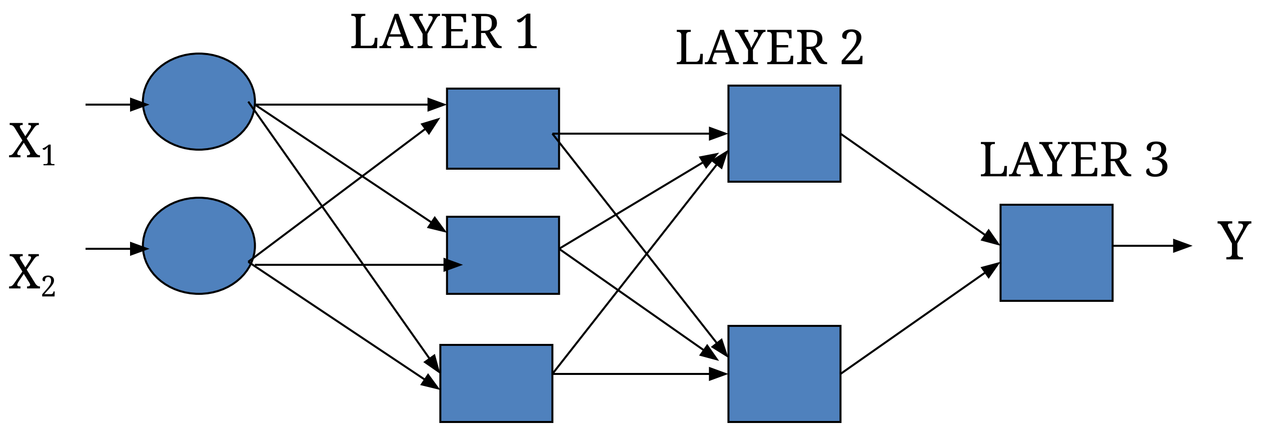
Cascading layers

Two layers :



Three layers:





Multi-layer network

Application for Change Detection

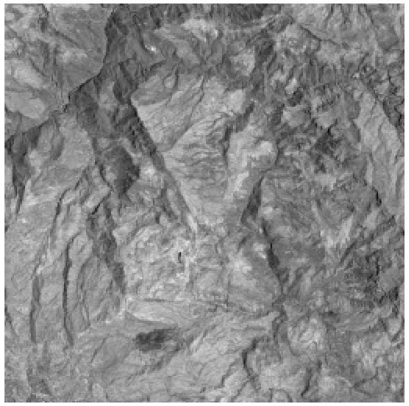


Image of date-1

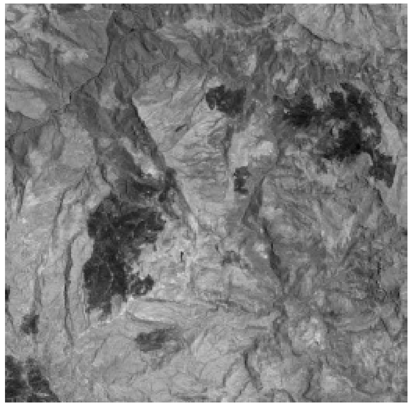
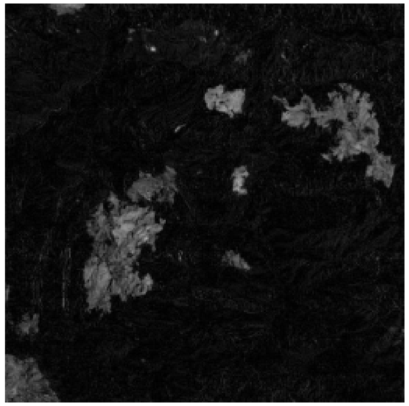
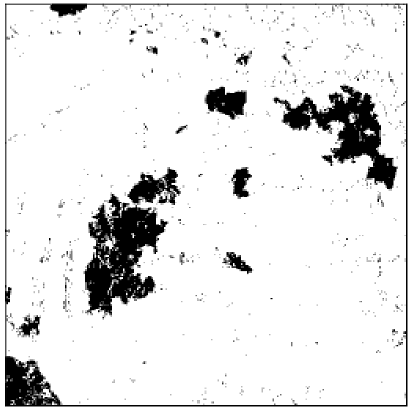
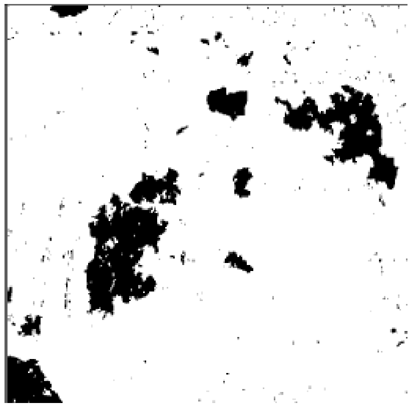
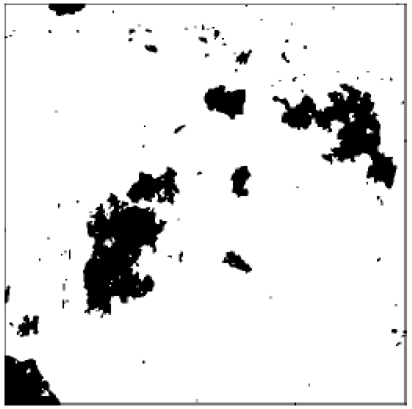


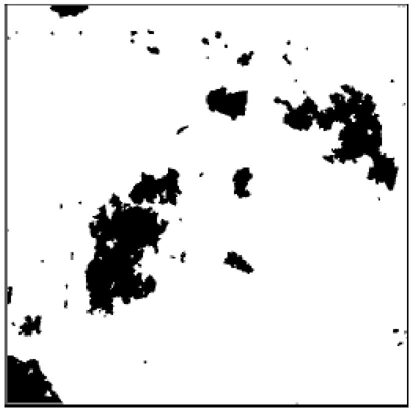
image of date-2


Difference image

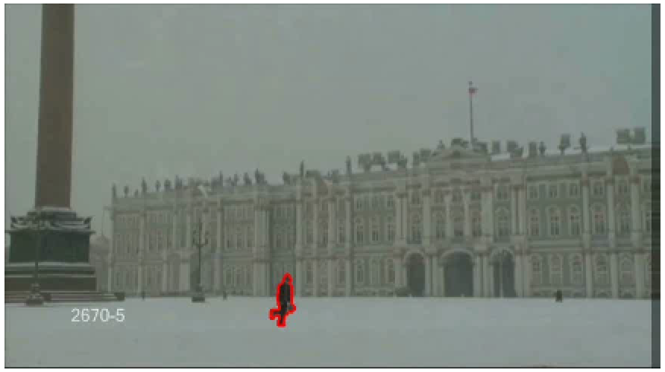

Min-distance


KNN


Bayes Classifier


Neural network

Video Tracking

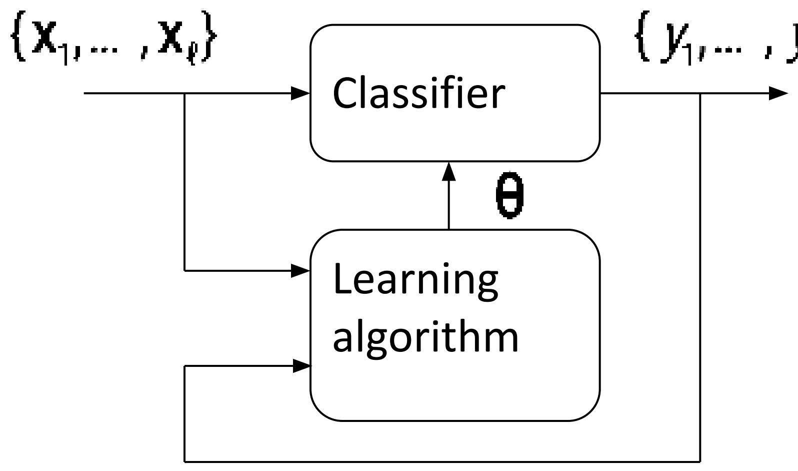


Unsupervised learning/ clustering

Input: Data{$\mathbf{x}_1$,…,$\mathbf{x}_\ell$} without information about the hidden state.

Clustering: goal is to find clusters of data sharing similar properties.

A broad class of unsupervised learning algorithms:



Classifier $q : X \times \Theta \to Y$

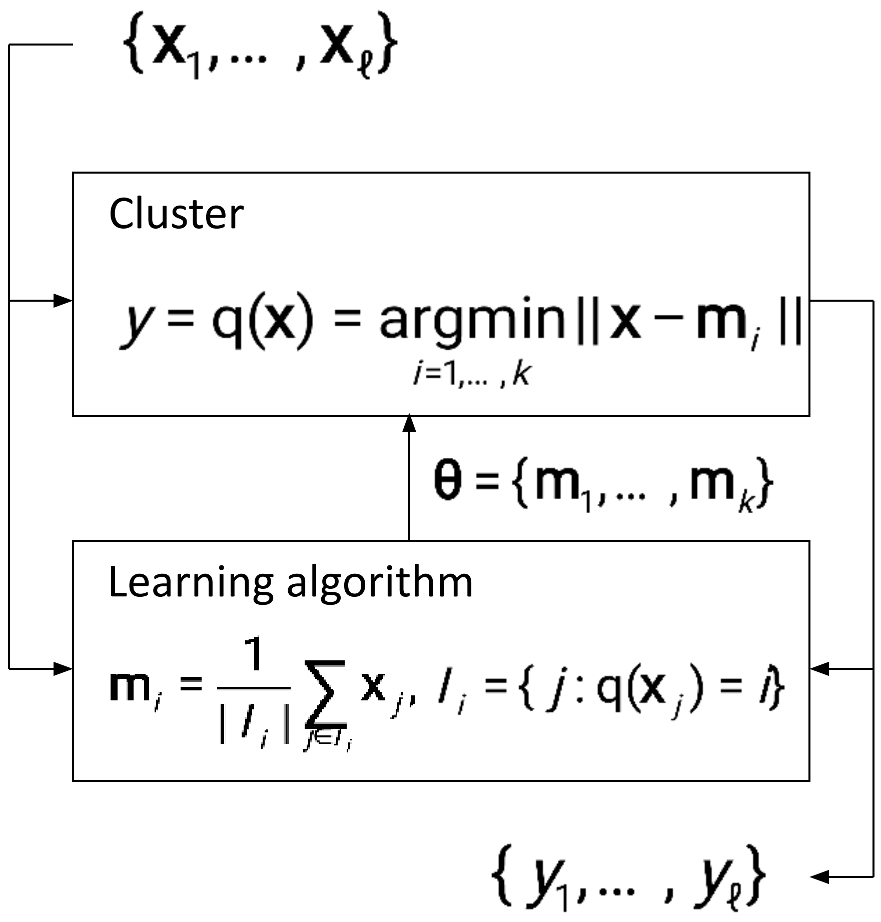
Learning algorithm (supervised) $L : (X \times Y)^\ell \to \Theta$

Good Clustering

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Example of unsupervised learning algorithm

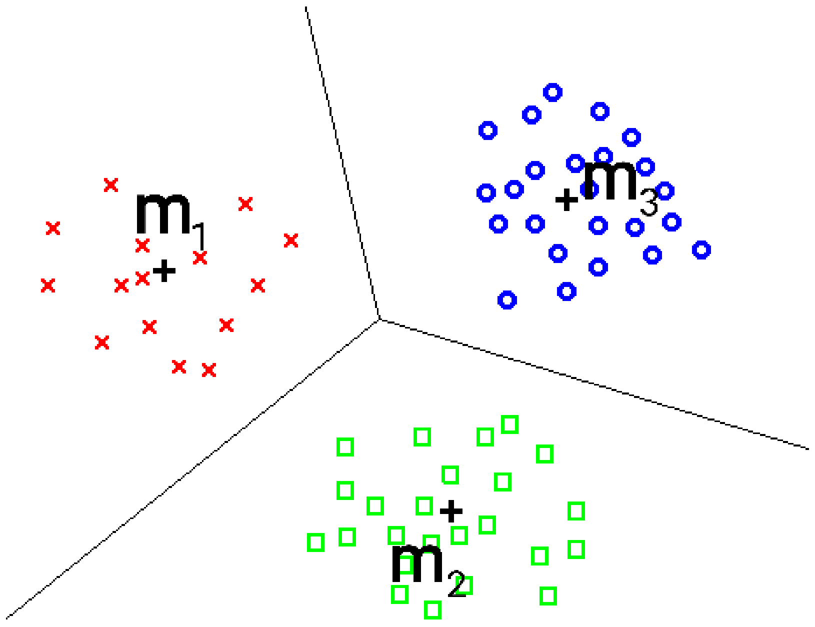
k-Means clustering:



$\{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$
Cluster
$y = q(\mathbf{x}) = \underset{i=1,\dots,k}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{m}_i\|$
$\theta = \{\mathbf{m}_1, \dots, \mathbf{m}_k\}$
Learning algorithm
$\mathbf{m}_i = \frac{1}{|I_i|} \sum_{j \in I_i} \mathbf{x}_j, \; I_i = \{ j : q(\mathbf{x}_j) = i \}$
$\{ y_1, \dots, y_\ell \}$

Goal is to minimize

$$\sum_{i=1}^{\ell} \|\mathbf{x}_i - \mathbf{m}_{q(\mathbf{x}_i)}\|^2$$



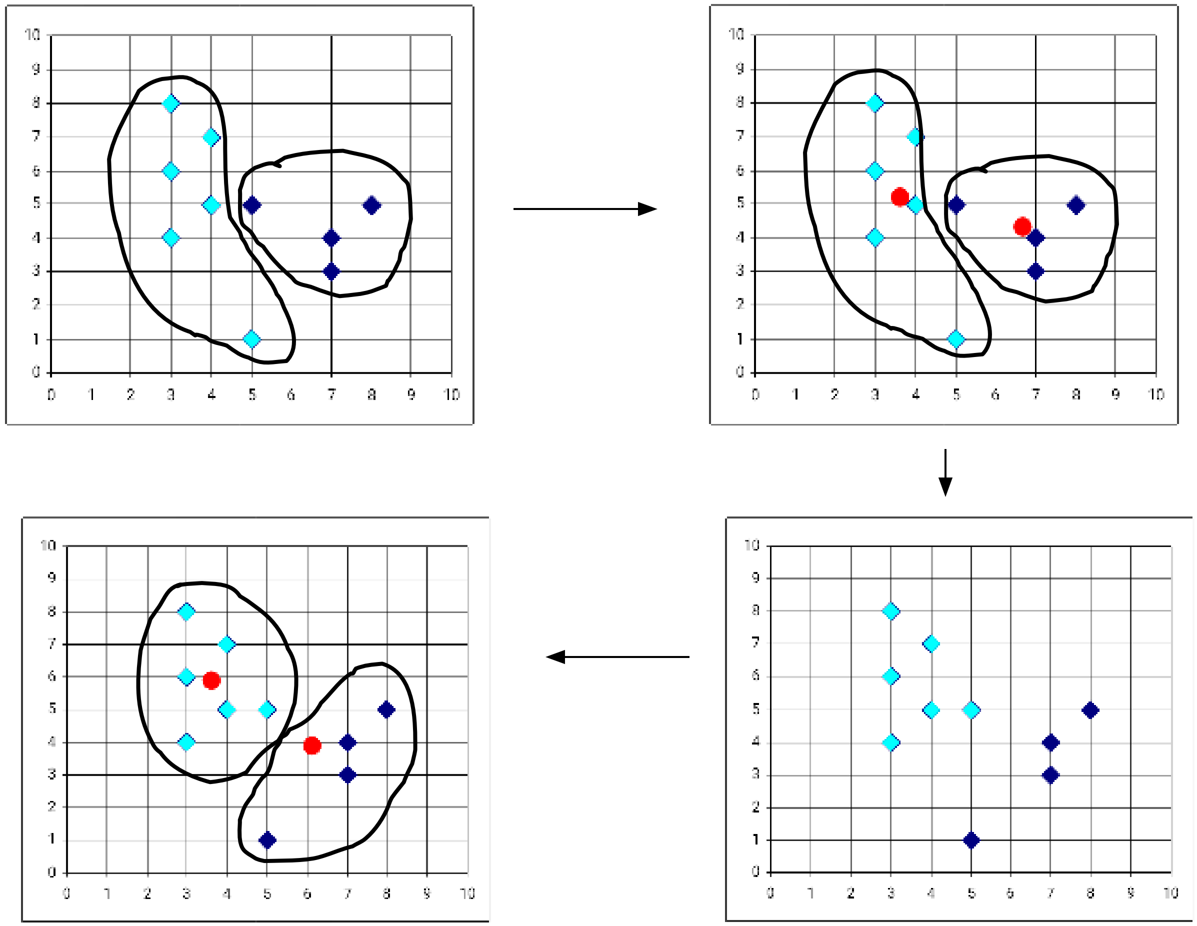
K-means clustering algorithm

Given *k*, the *k-means* algorithm is implemented in 4 steps:

i. Partition objects into *k* nonempty subsets
ii. Compute seed points as the centroids of the clusters of the current partition.
iii. The centroid is the center (mean point) of the cluster.
iv. Assign each object to the cluster with the nearest seed point.
v. Go back to Step iii, stop when no more new assignment.

K-means clustering algorithm

- Example



References

Books

Theodoridis and Koutroumbas, Pattern Recognition. Academic Press, San Diego, 2003.

Duda and Heart: Pattern Classification and Scene Analysis. J. Wiley & Sons, New York, 1982. (2nd edition 2000).

Fukunaga: Introduction to Statistical Pattern Recognition. Academic Press, 1990.

Bishop: Neural Networks for Pattern Recognition. Claredon Press, Oxford, 1997.

Thank you for your attention!

“But before we move on, allow me to belabor the point even further...”

Questions?