

# Image Processing

## Lecture 15: Image Pattern Classification (Ch12 Image Pattern Classification)

Zhiguo Zhang

[zhiguo Zhang@hit.edu.cn](mailto:zhiguo Zhang@hit.edu.cn)



# Review of Last Lecture

---

- In the last lecture we learnt:
  - Background of feature extraction
  - Boundary feature descriptor
  - Region feature descriptor
  - Other descriptors

# Contents of This Lecture

---

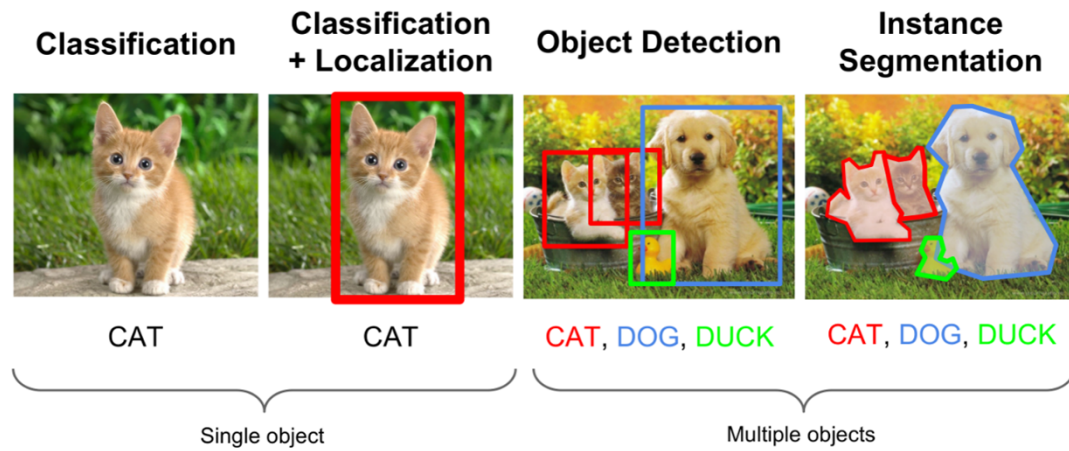
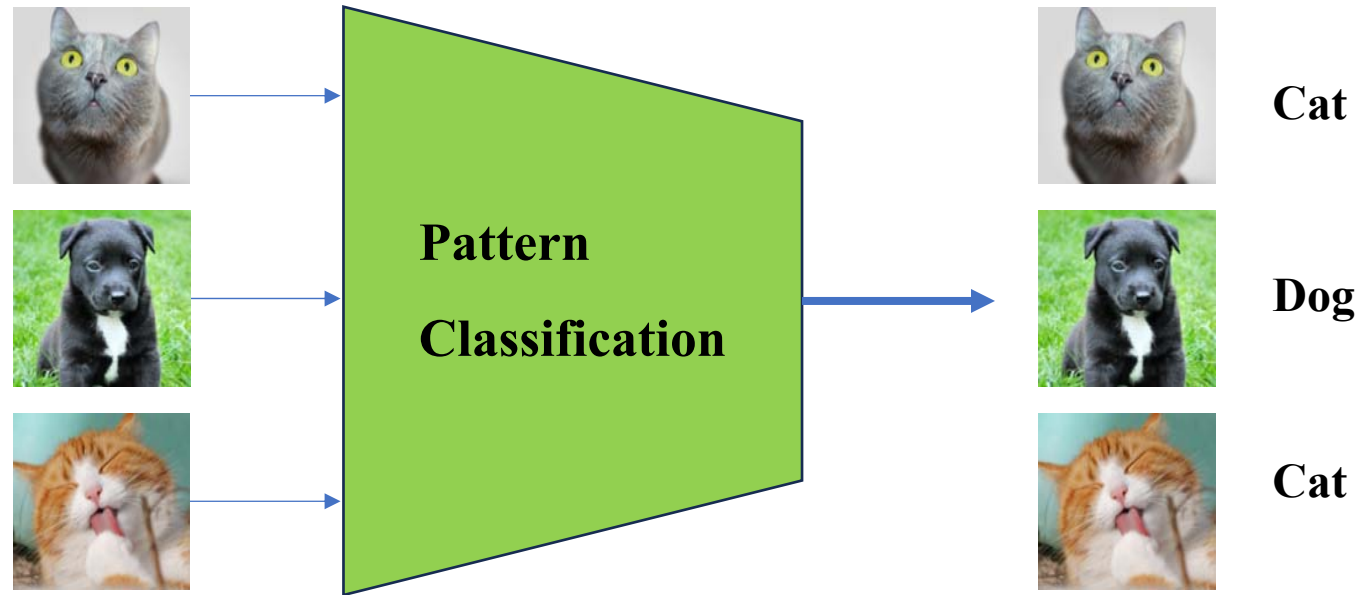
- Background of Pattern Classification
- Patterns and Pattern Classes
- Pattern Classification by Prototype Matching
- Optimal (Bayes) Statistical Classifiers
- Neural Networks and Deep Learning
- Convolutional Neural Networks

# Background

---

- In image pattern recognition, we think of a *pattern* as a spatial arrangement of features.
- A *pattern class* is a set of patterns that share some common properties.
- Pattern recognition encompasses techniques for automatically assigning patterns to their respective classes.
- That is, given a pattern or sets of patterns whose class is unknown, the job of a pattern recognition system is to assign a class label to each of its input patterns.

# Background



# Background

---

- There are four main stages involved in recognition:
  - 1) **sensing**: generate signals in a spatial (2-D) or higher-dimensional format
  - 2) **preprocessing**: perform tasks such as noise reduction, enhancement, restoration, and segmentation
  - 3) **feature extraction**: detect and extract quantitative measures for labelling and discrimination
  - 4) **classification**: assign class labels to unknown input image patterns

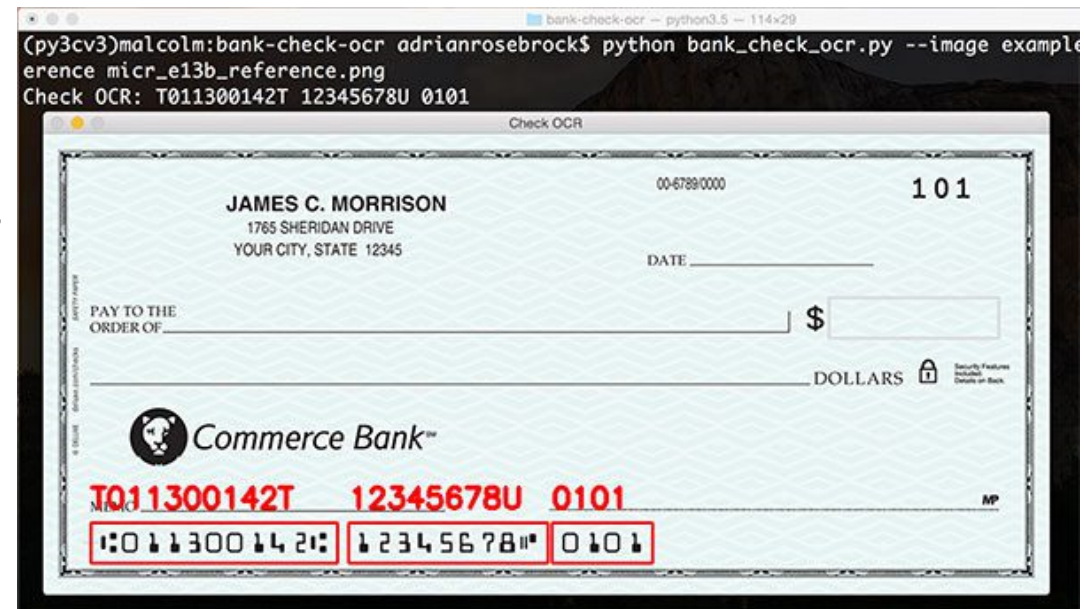
# Background

---

- Three basic approaches used for image pattern classification:
  1. classification based on matching unknown patterns against specified prototypes
  2. optimal statistical classifiers
  3. neural networks
- Ultimately, recognition performance is determined by the discriminative power of the features used.

# Background

- In classification based on prototypes, the objective is to make the features so unique and easily detectable that classification itself becomes a simple task.
- A good example of this are bank-check processors, which use stylized font styles to simplify machine processing (the example will be shown later).





# Background

---

- In the second category (optimal statistical classifiers), classification is cast in decision-theoretic, statistical terms.
- The classification approach is based on selecting parameters that can be shown to yield optimal classification performance in a statistical sense.
- Here, emphasis is placed on both the features used, and the design of the classifier.

# Background

---

- In the third category, classification is performed using neural networks.
- Neural networks can operate using engineered features too, but they have the unique ability of being able to generate, on their own, representations (features) suitable for recognition.
- These systems can accomplish this using raw data, without the need for engineered features.

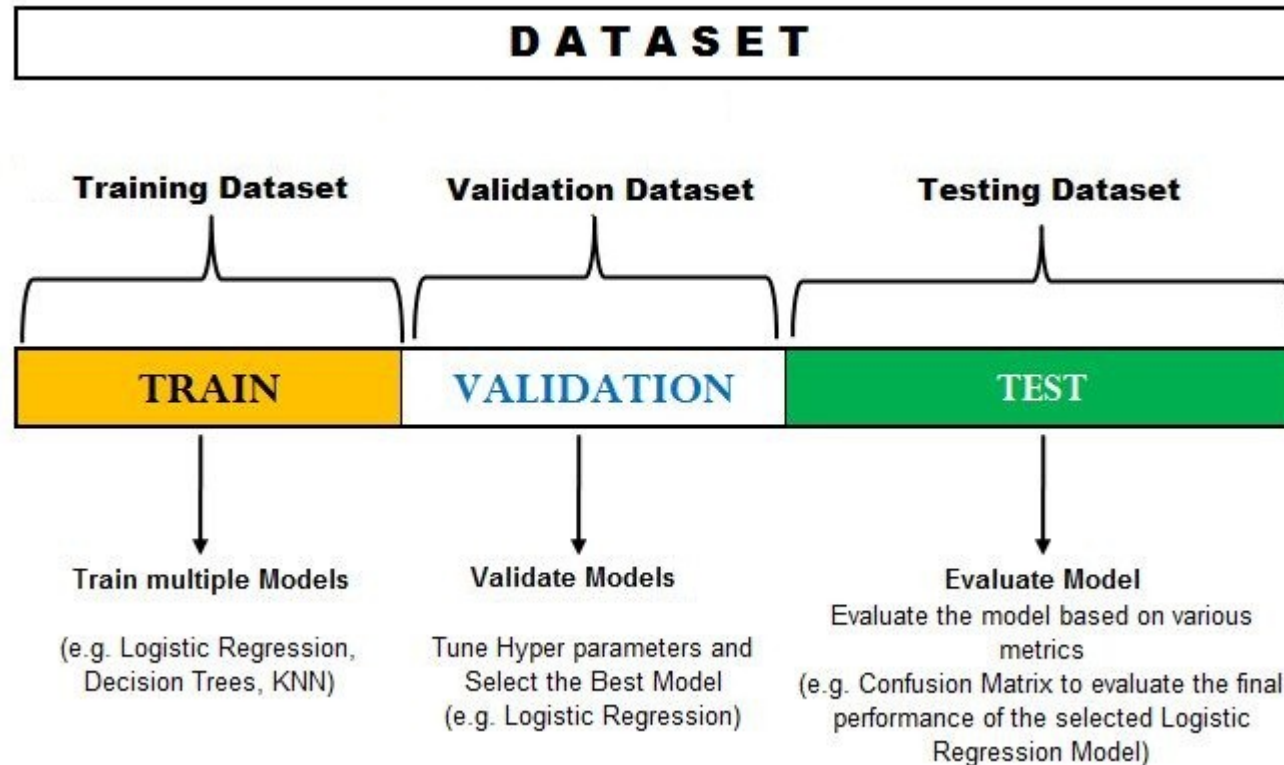
# Background

---

- All three approaches are based on parameters that must be either specified or learned from patterns that represent the recognition problem we want to solve.
- The patterns can be *labeled*, meaning that we know the class of each pattern, or *unlabeled*, meaning that the data are known to be patterns, but the class of each pattern is unknown.

# Background

- When working with a labeled data, a given data set generally is subdivided into three subsets: *a training set, a validation set, and a test set.*
  - A typical subdivision might be 50% training, and 25% each for the validation and test sets.



# Background

---

- The process by which a training set is used to generate classifier parameters is called *training*.
- **Training:** In this mode, a classifier is given the class label of each pattern, the objective being to make adjustments in the parameters if the classifier makes a mistake in identifying the class of the given pattern.

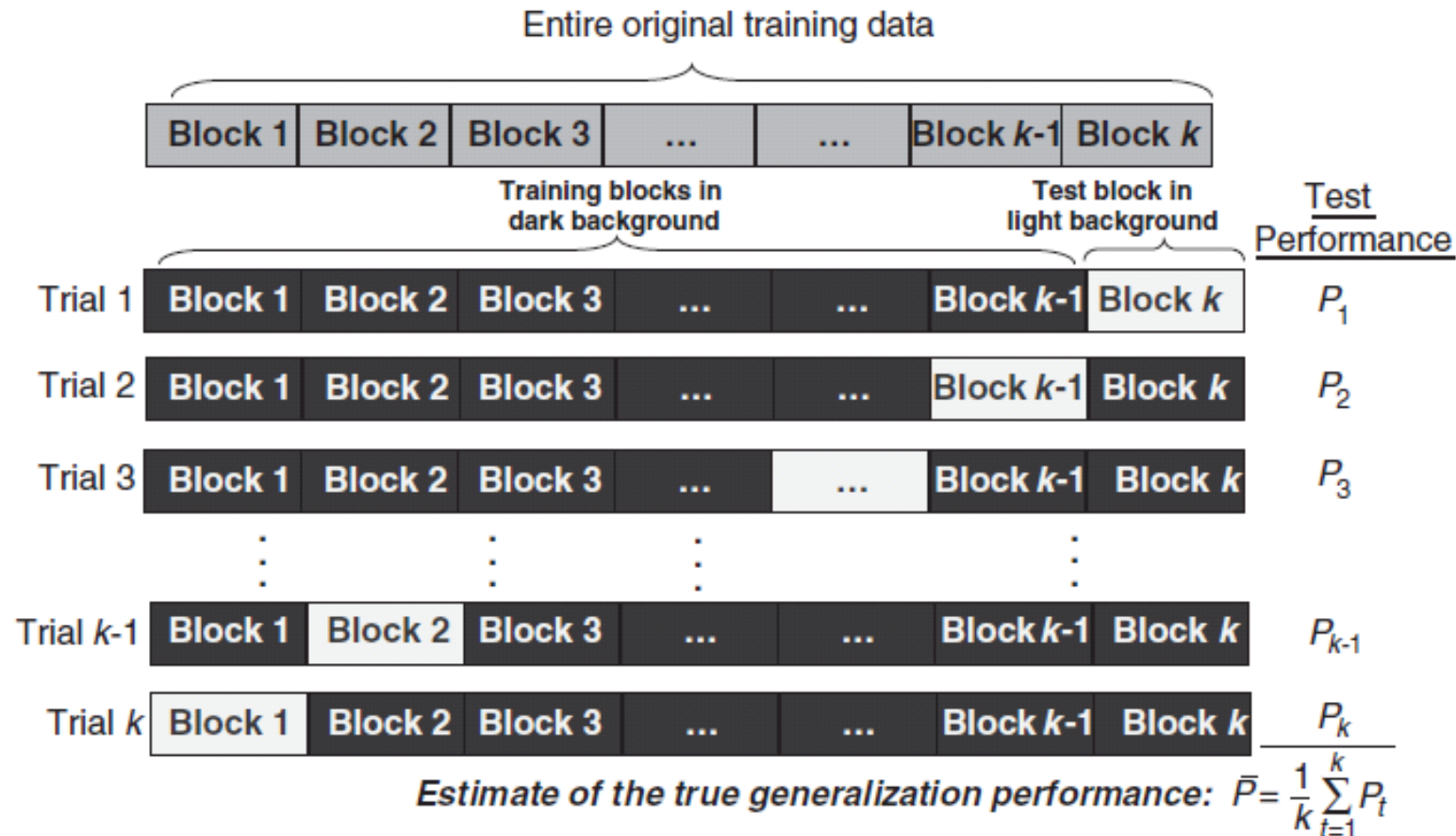
# Background

---

- **Validation:** At this point, we might be working with several candidate designs. At the end of training, we use the *validation* set to compare the various designs against a performance objective.
- Typically, several iterations of training/validation are required to establish the design that comes closest to meeting the desired objective.

# Background

- **Cross Validation (CV):** to perform multiple trials of validation using different partitions, and to average the results over the rounds.



# Background

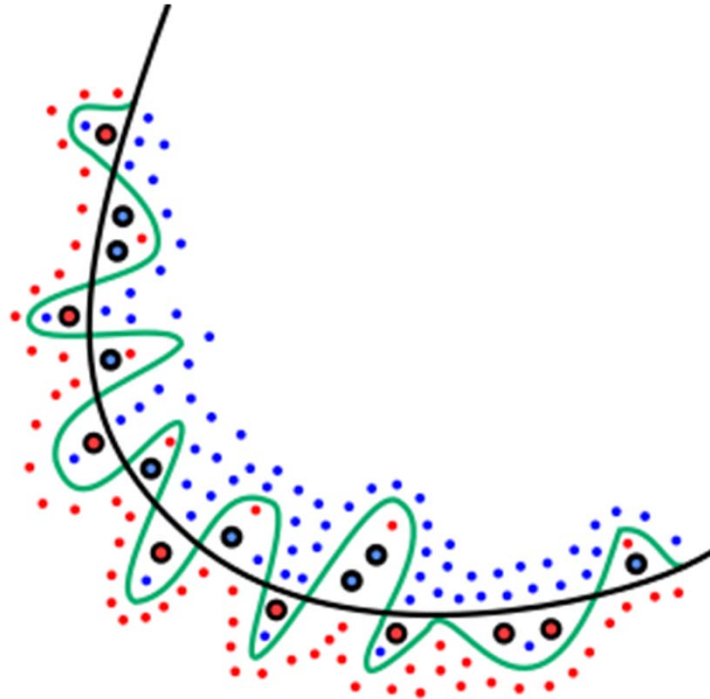
---

- **Test:** Once a design has been selected, the final step is to determine how it will perform “in the field.” For this, we use the test set, which consists of patterns that the system has never “seen” before.
- If the training and validation sets are truly representative of the data the system will encounter in practice, the results of training/validation should be close to the performance using the test set.



# Background

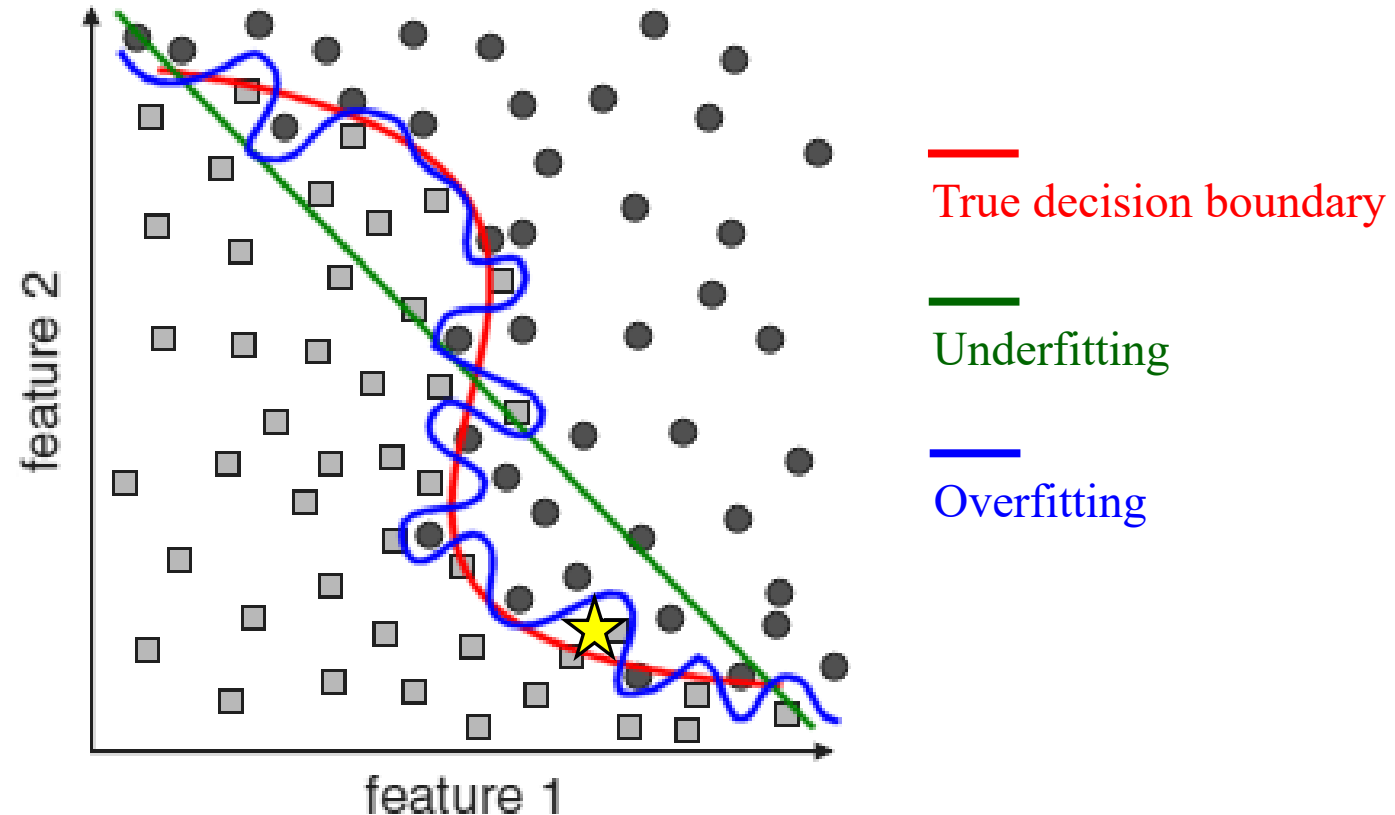
- If training/validation results are acceptable, but test results are not, we say that training/validation “**over fit**” the system parameters to the available data.
- Over-fitting is often caused by a too complicated system.



The green line represents an overfitted model and the black line represents a regularized model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen test data illustrated by black-outlined dots, compared to the black line.

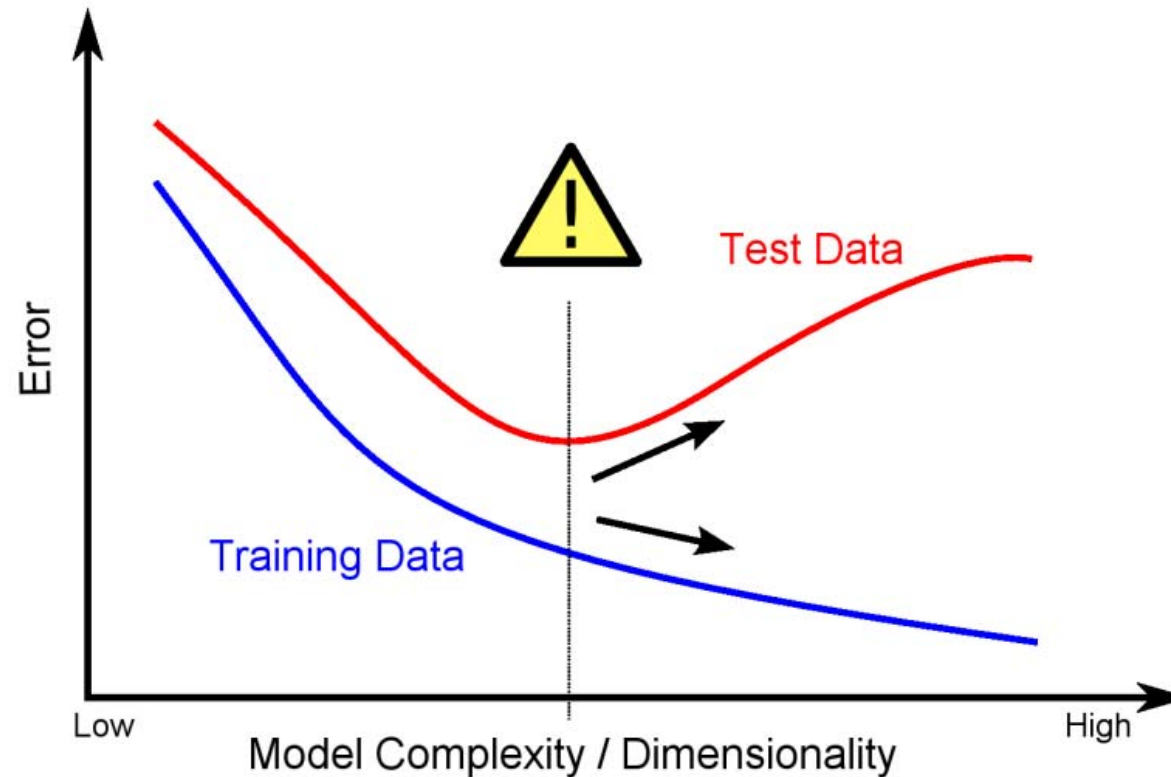
# Background

- Overfitting (the classifier is too complicated) and underfitting (the classifier is too simple).



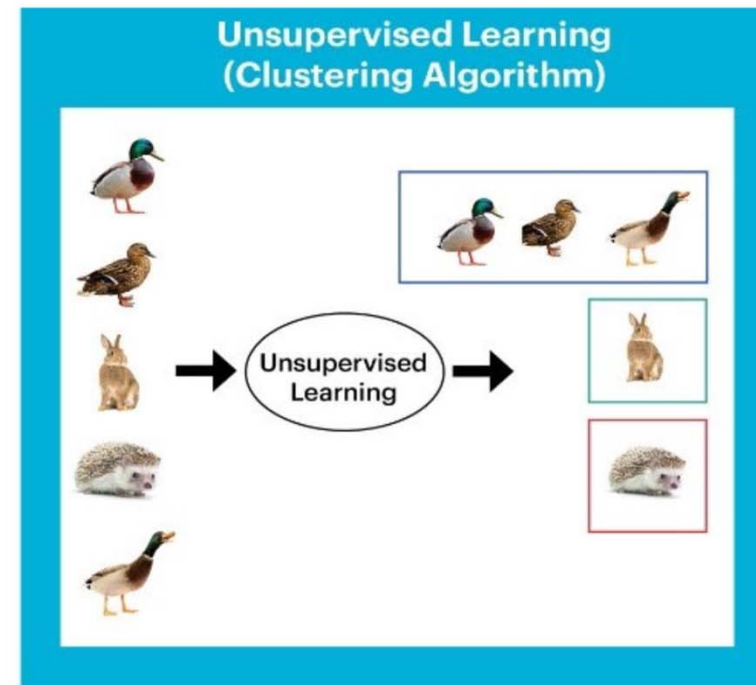
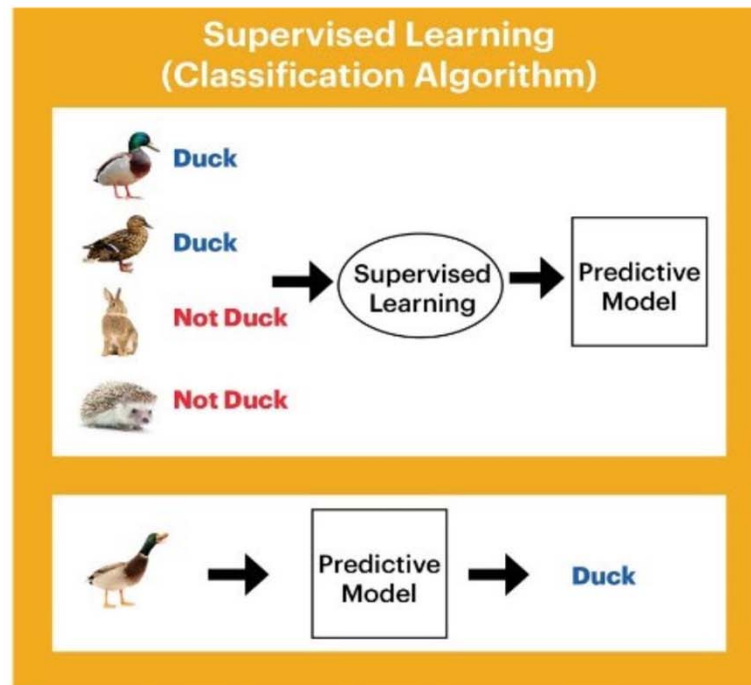
# Background

- The best model would be where the error on test data has its global minimum.



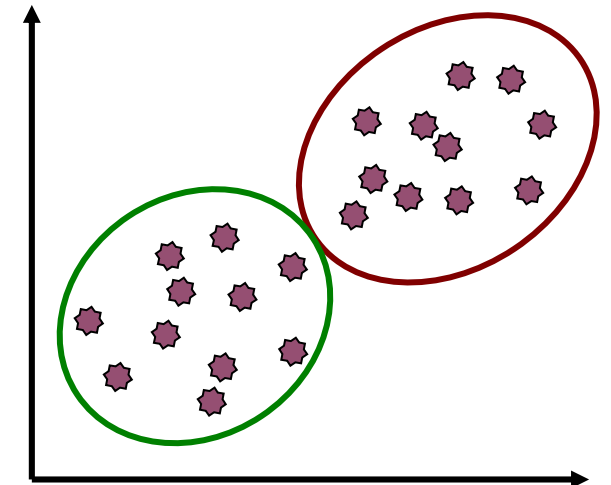
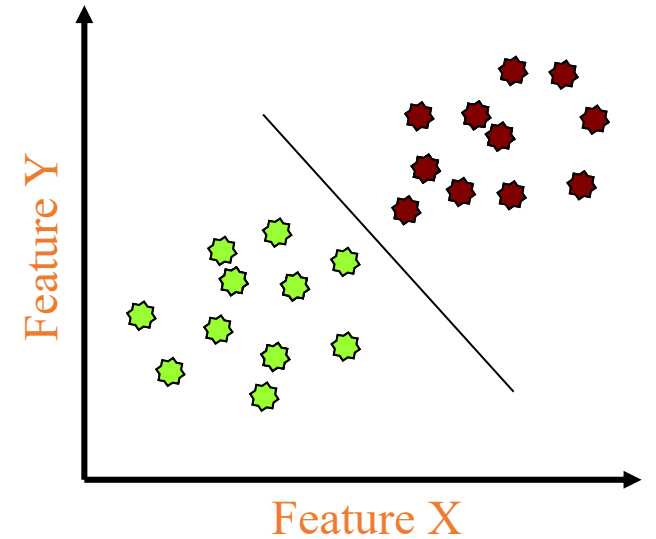
# Background

- A system that is designed using *labeled* training data is said to undergo *supervised learning (classification)*.
- A system learns the pattern classes themselves with *unlabeled* data is said to undergo *unsupervised learning (clustering)*.



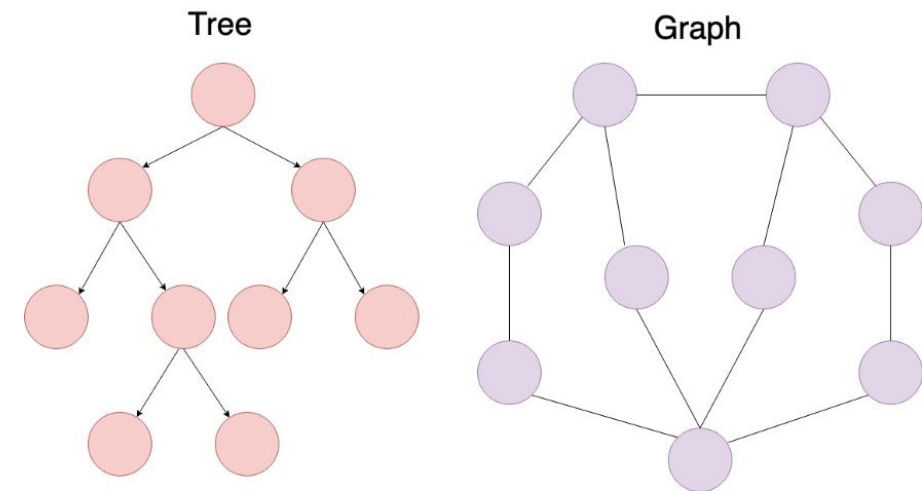
# Background

- Classification
  - Have labels for some samples
  - Design a “rule” to accurately assign labels to new samples
  - Supervised learning
- Clustering
  - No labels
  - Group samples into clusters based on how “near” they are to one another
  - Unsupervised learning



# Patterns and Pattern Classes

- For the purpose of recognition, we expect that **pattern classes to be grouped tightly, and as far away from each other as possible.**
- In image pattern classification, two principal pattern arrangements are quantitative and structural.
- *Quantitative patterns* are arranged in the form of *pattern vectors*.
- *Structural patterns* typically are composed of symbols, arranged in the form of *strings*, *trees*, or *graphs*.



# Pattern Vectors

---

- Pattern vectors:

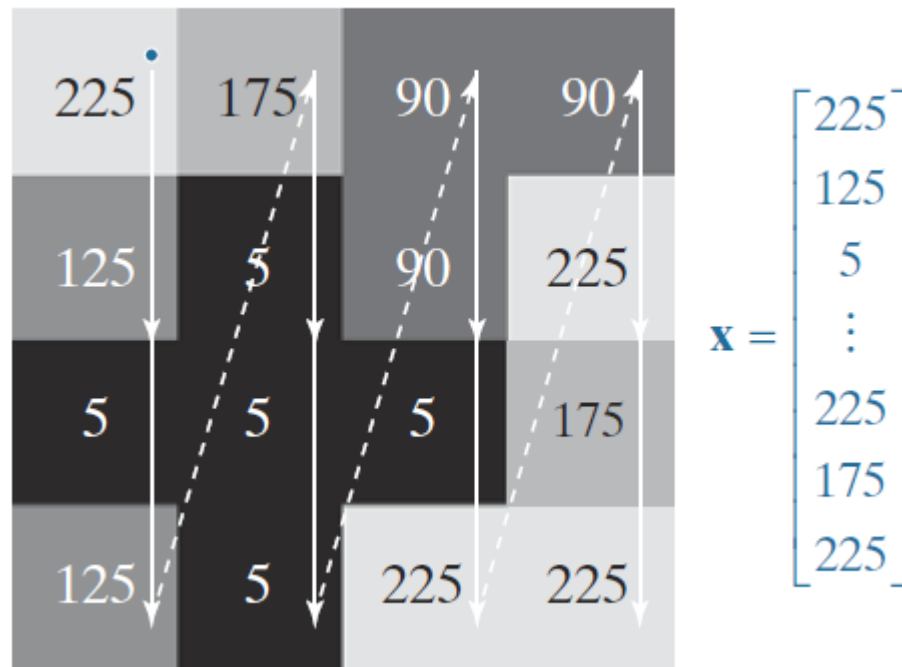
$$\mathbf{x} = [x_1, x_2, \dots, x_n]^T$$

where each component,  $x_i$ , represents the  $i$ th feature descriptor, and  $n$  is the total number of such descriptors.

- A pattern vector may be “viewed” as a point in  $n$ -dimensional Euclidean space, and a pattern class may be interpreted as a “hypercloud” of points in this *pattern space*.

# Pattern Vectors

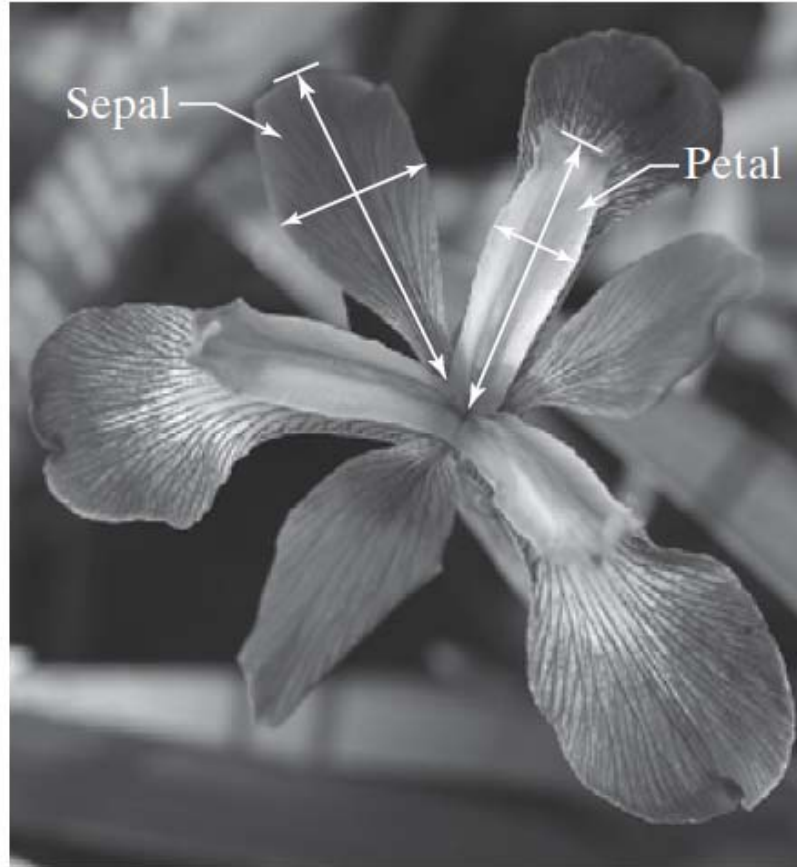
- Pattern vectors can be formed directly from image pixel intensities by vectorizing the image using, for example, linear indexing.





# Pattern Vectors

Petal and sepal width and length measurements (see arrows) performed on iris flowers for the purpose of data classification.

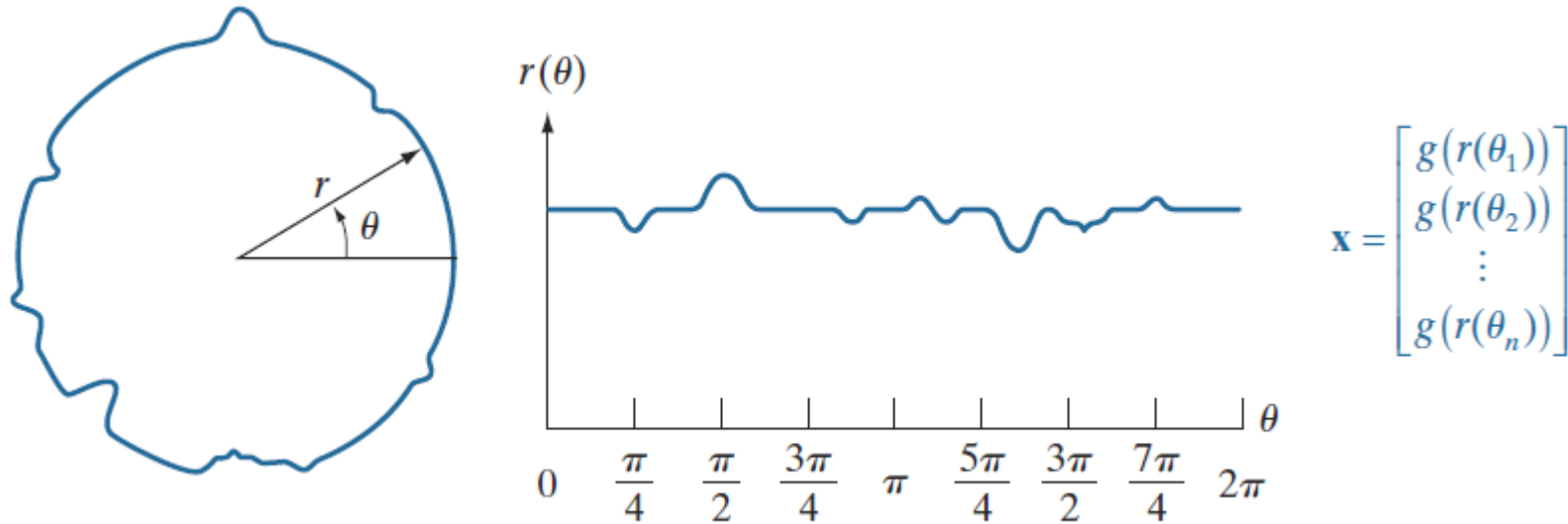


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$x_1$  = Petal width  
 $x_2$  = Petal length  
 $x_3$  = Sepal width  
 $x_4$  = Sepal length

# Pattern Vectors

- A higher-level representation of patterns is based on feature descriptors of the types we learned in Lecture 14.



A noisy object boundary, and its corresponding signature.

# Pattern Vectors

- A higher-level representation of patterns is based on feature descriptors of the types we learned in Lecture 14.

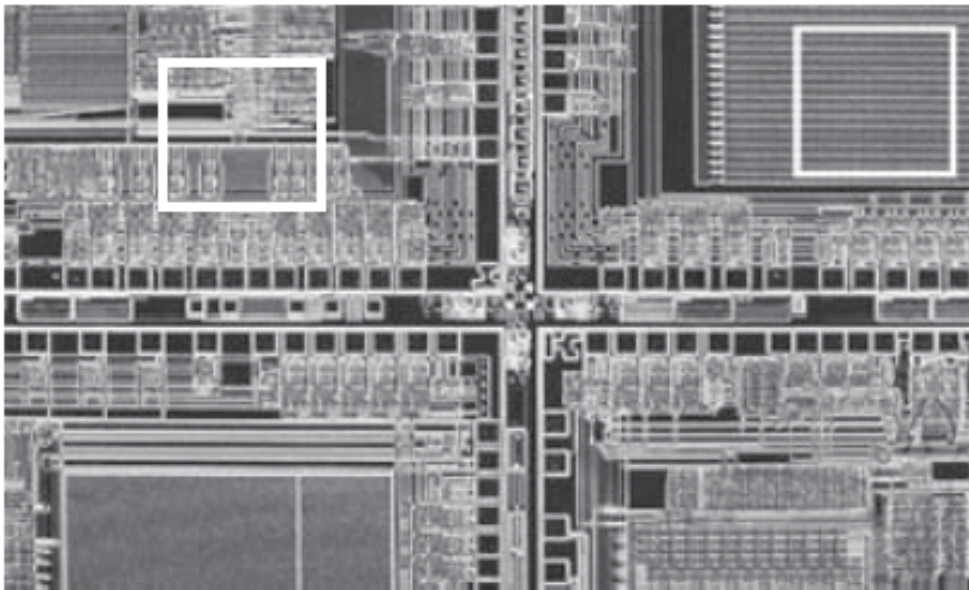


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \begin{array}{l} x_1 = \text{compactness} \\ x_2 = \text{circularity} \\ x_3 = \text{eccentricity} \end{array}$$

Pattern vectors whose components capture both boundary and regional characteristics.

# Pattern Vectors

- A higher-level representation of patterns is based on feature descriptors of the types we learned in Lecture 14.



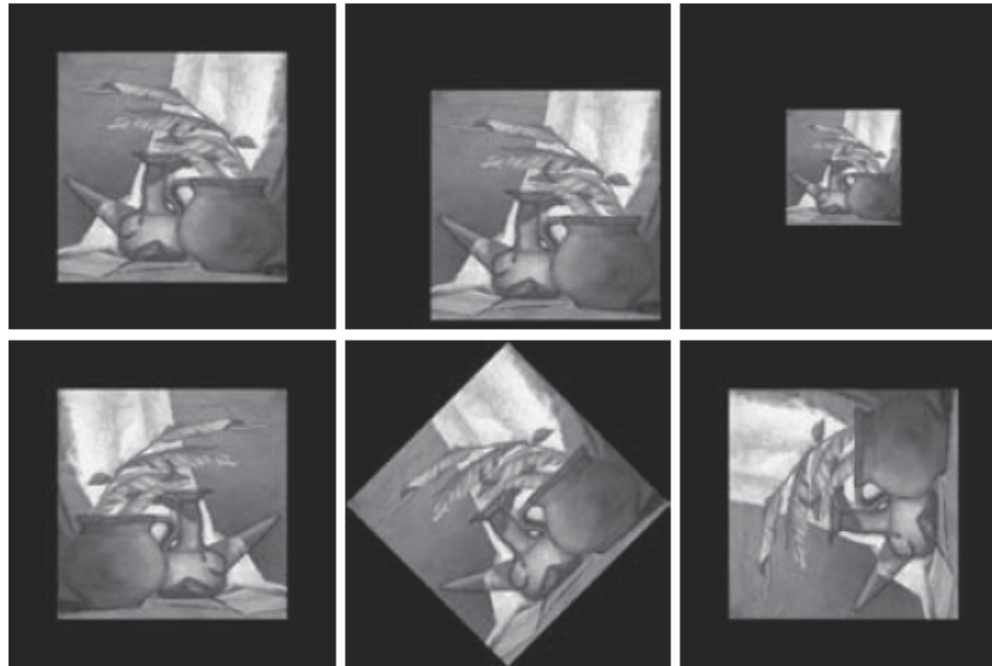
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

$x_1$  = max probability  
 $x_2$  = correlation  
 $x_3$  = contrast  
 $x_4$  = uniformity  
 $x_5$  = homogeneity  
 $x_6$  = entropy

An example of pattern vectors based on properties of gray level co-occurrence matrix of subimages.

# Pattern Vectors

- A higher-level representation of patterns is based on feature descriptors of the types we learned in Lecture 14.



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \end{bmatrix}$$

The  $\phi$ 's are moment invariants

Feature vectors with components that are invariant to transformations such as rotation, scaling, and translation. The vector components are moment invariants.

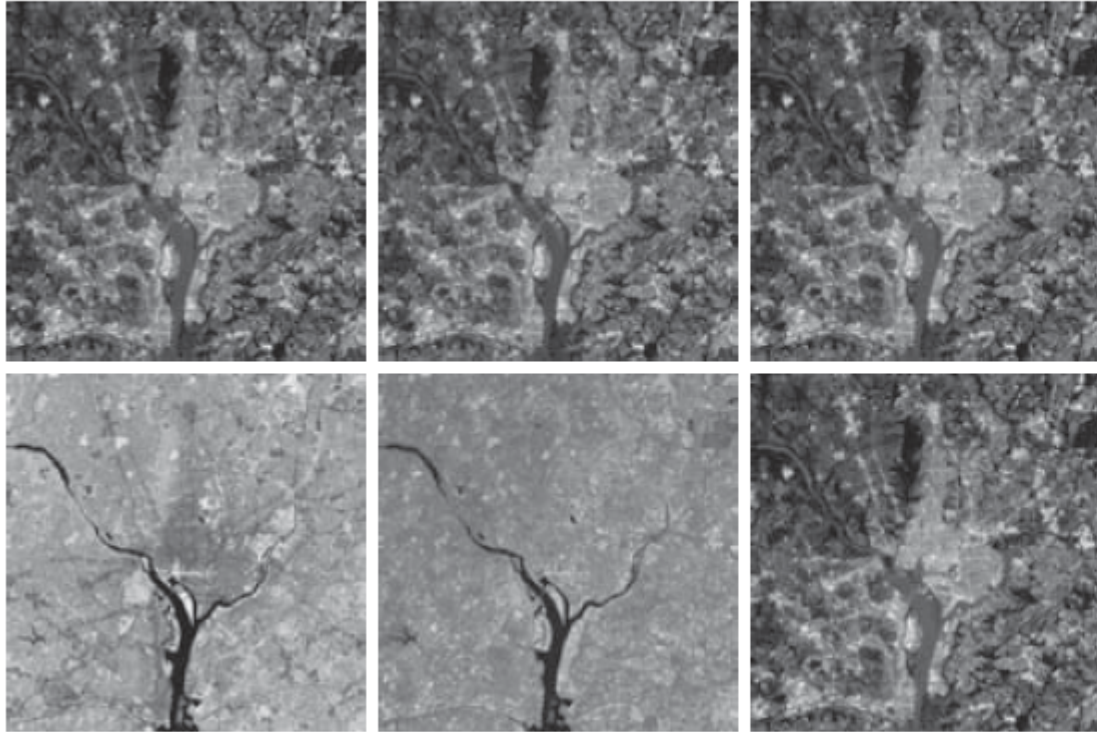
# Pattern Vectors

---

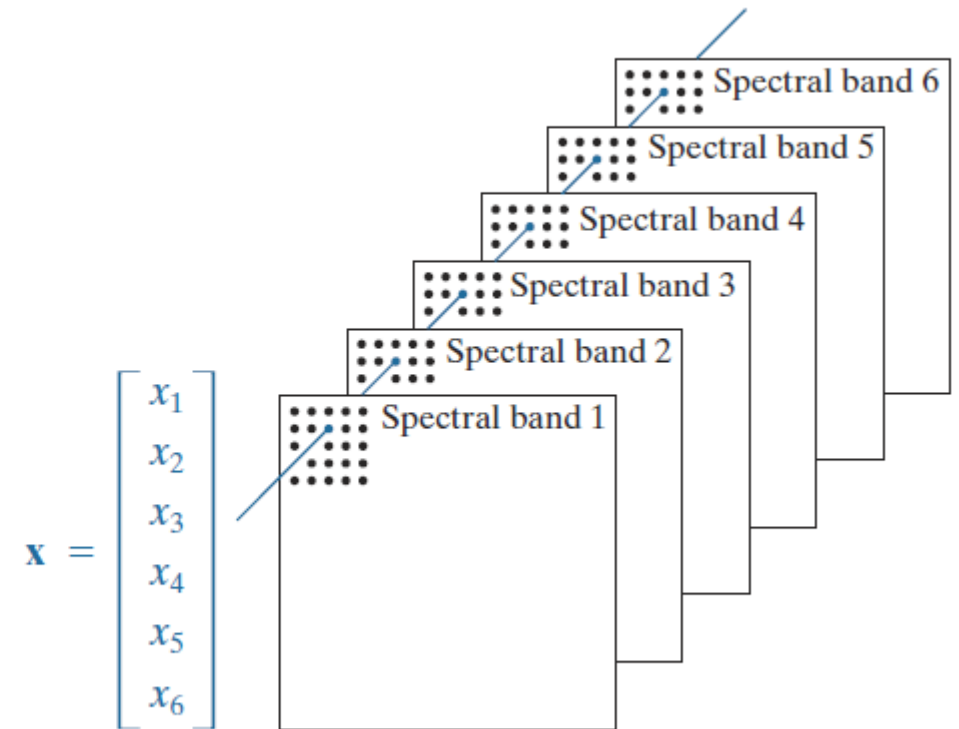
- When working with sequences of registered images (different sets of images are transformed into one coordinate system), we have the option of using pattern vectors formed from corresponding pixels in those images.
- Forming pattern vectors in this way implies that recognition will be based on information extracted from the same spatial location across the images.
- It is ideally suited for applications such as recognizing regions in multispectral images,

# Pattern Vectors

Images in spectral bands 1–3



Images in spectral bands 4–6



Pattern (feature) vectors formed by concatenating corresponding pixels from a set of registered images

# Structural Patterns

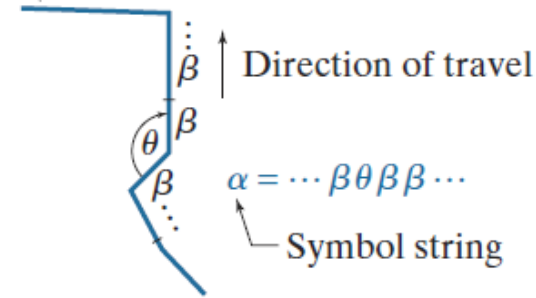
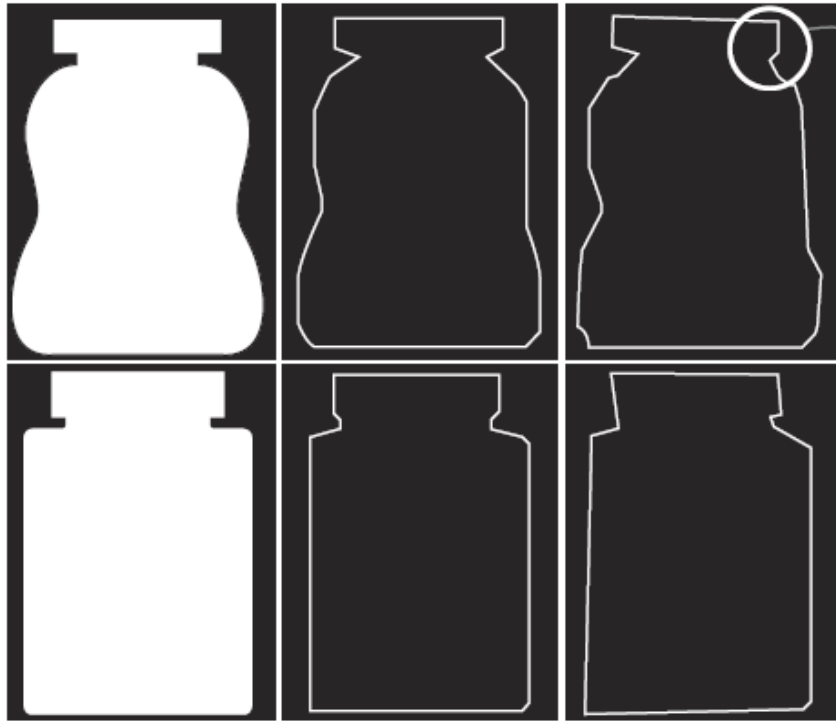
---

- Pattern vectors are not suitable for applications in which objects are represented by **structural features**, such as strings of symbols, tree, and graph.
- Although they are used much less than vectors in image processing applications, patterns containing structural descriptions of objects are important in applications where shape is of interest.



# Structural Patterns

- String of symbols

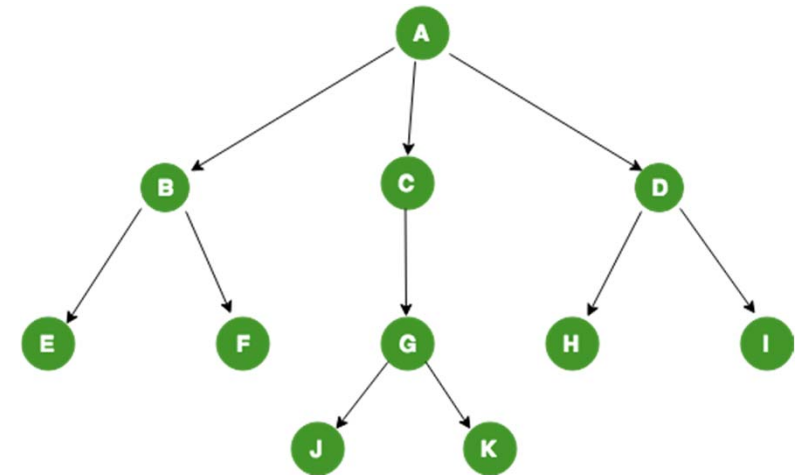
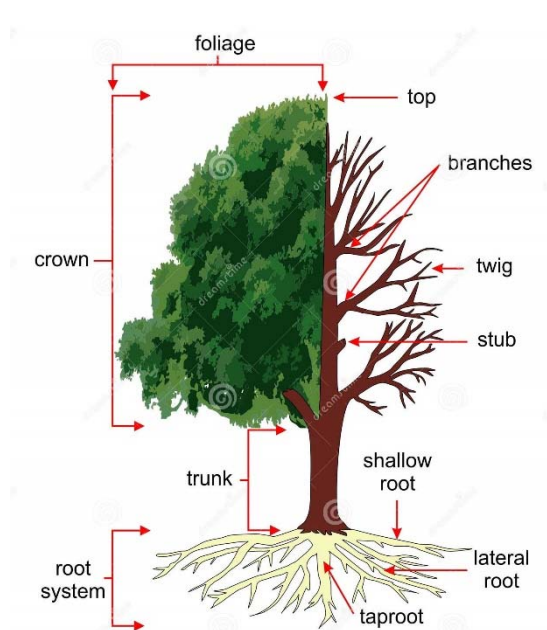


$\theta$  = interior angle  
 $\beta$  = line segment of specified length

- The boundaries of the bottles were approximated by a polygon.
- The boundary is subdivided into line segments ( $\beta$ ), and the interior angle,  $\theta$ , is computed at each intersection of two line segments.
- A string of sequential symbols is generated as the boundary is traversed in the counterclockwise direction.
- Strings of this form are structural patterns, and the objective is to match a given string against stored string prototypes.

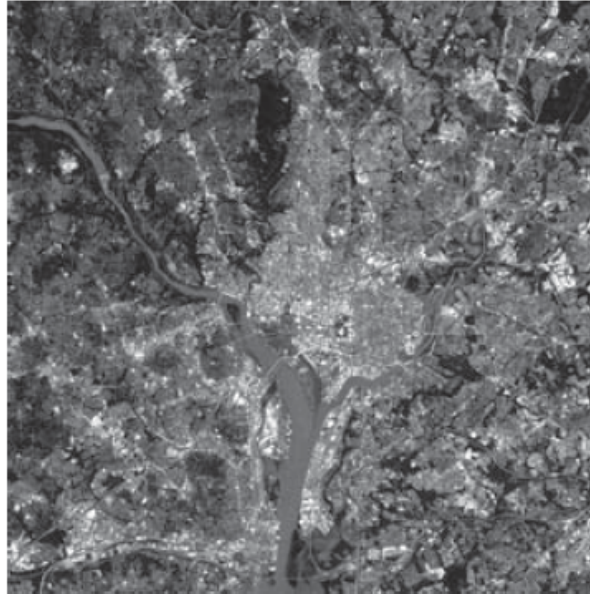
# Structural Patterns

- A *tree* is another structural representation, suitable for higher-level descriptions of an entire image in terms of its component regions.
- Basically, most hierarchical ordering schemes lead to tree structures.

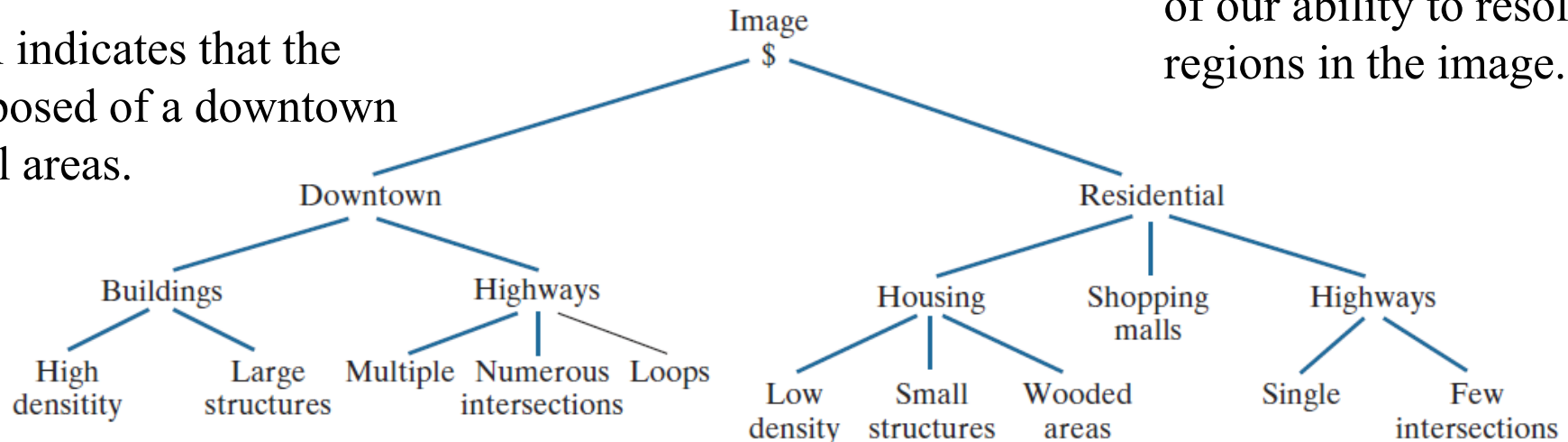


# Structural Patterns

- A satellite image of a heavily built downtown area and surrounding residential areas.
- The (upside down) tree was obtained using the structural relationship “composed of.”
- Thus, the root of the tree represents the entire image.
- The next level indicates that the image is composed of a downtown and residential areas.



- In turn, the residential areas are composed of housing, highways, and shopping malls.
- The next level down in the tree further describes the housing and highways.
- We can continue this type of subdivision until we reach the limit of our ability to resolve different regions in the image.



# Prototype Matching

---

- Prototype matching involves comparing an unknown pattern against a set of prototypes, and assigning to the unknown pattern the class of the prototype that is the most “similar” to the unknown.
- Each prototype represents a unique pattern class, but there may be more than one prototype for each class.
- What distinguishes one matching method from another is the measure used to determine similarity.

# Minimum Distance Classifier

---

- The simplest approach is the **minimum distance classifier**.
- It calculates a distance-based measure between an unknown pattern vector and each of the class prototypes.
- Then it assigns the unknown pattern to the class of its closest prototype.

# Minimum Distance Classifier

---

- The prototype vectors of the minimum-distance classifier usually are the mean vectors of the various pattern classes:

$$\mathbf{m}_j = \frac{1}{n_j} \sum_{\mathbf{x} \in c_j} \mathbf{x} \quad j = 1, 2, \dots, N_c$$

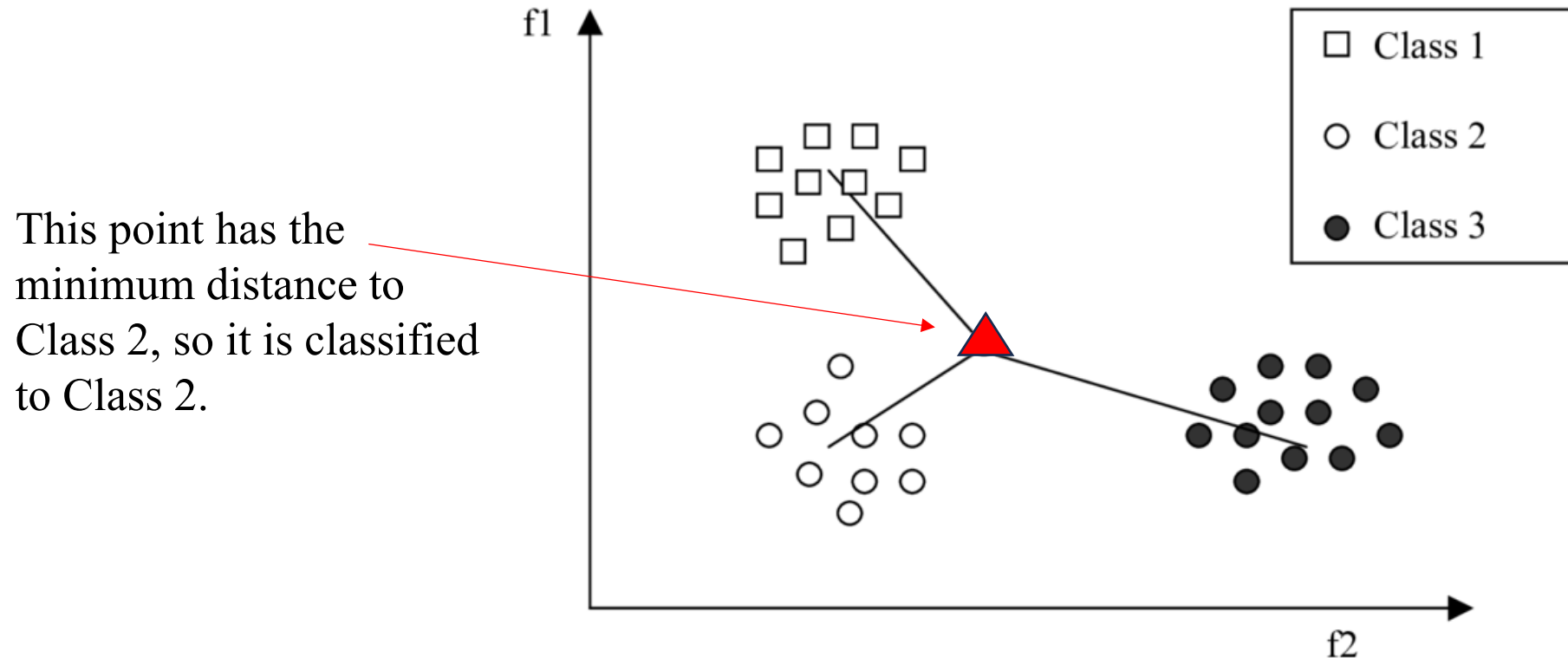
where  $n_j$  is the number of pattern vectors used to compute the  $j$ th mean vector,  $c_j$  is the  $j$ th pattern class, and  $N_c$  is the number of classes.

- If we use the Euclidean distance to determine similarity, the minimum-distance classifier computes the distances

$$D_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_j\| = \left[ (\mathbf{x} - \mathbf{m}_j)^T (\mathbf{x} - \mathbf{m}_j) \right]^{1/2} \quad j = 1, 2, \dots, N_c$$

# Minimum Distance Classifier

- The classifier then assigns an unknown pattern  $\mathbf{x}$  to class  $c_i$  if  $D_i(\mathbf{x}) < D_j(\mathbf{x})$  for  $j = 1, 2, \dots, N_c, j \neq i$ .



# Minimum Distance Classifier

---

- Minimizing the square of  $D_j(\mathbf{x})$  leads to

$$\begin{aligned} D_j^2(\mathbf{x}) &= (\mathbf{x} - \mathbf{m}_j)^T (\mathbf{x} - \mathbf{m}_j) = \mathbf{x}^T \mathbf{x} - 2\mathbf{x}^T \mathbf{m}_j + \mathbf{m}_j^T \mathbf{m}_j \\ &= \mathbf{x}^T \mathbf{x} - 2\left(\mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j\right) \end{aligned}$$

- Note that the term  $\mathbf{x}^T \mathbf{x}$  is independent of  $j$  (that is, it is a constant with respect to  $j$ ).
- Thus, choosing the minimum of  $D_j(\mathbf{x})$  is equivalent to choosing the maximum of  $\left(\mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j\right)$ .



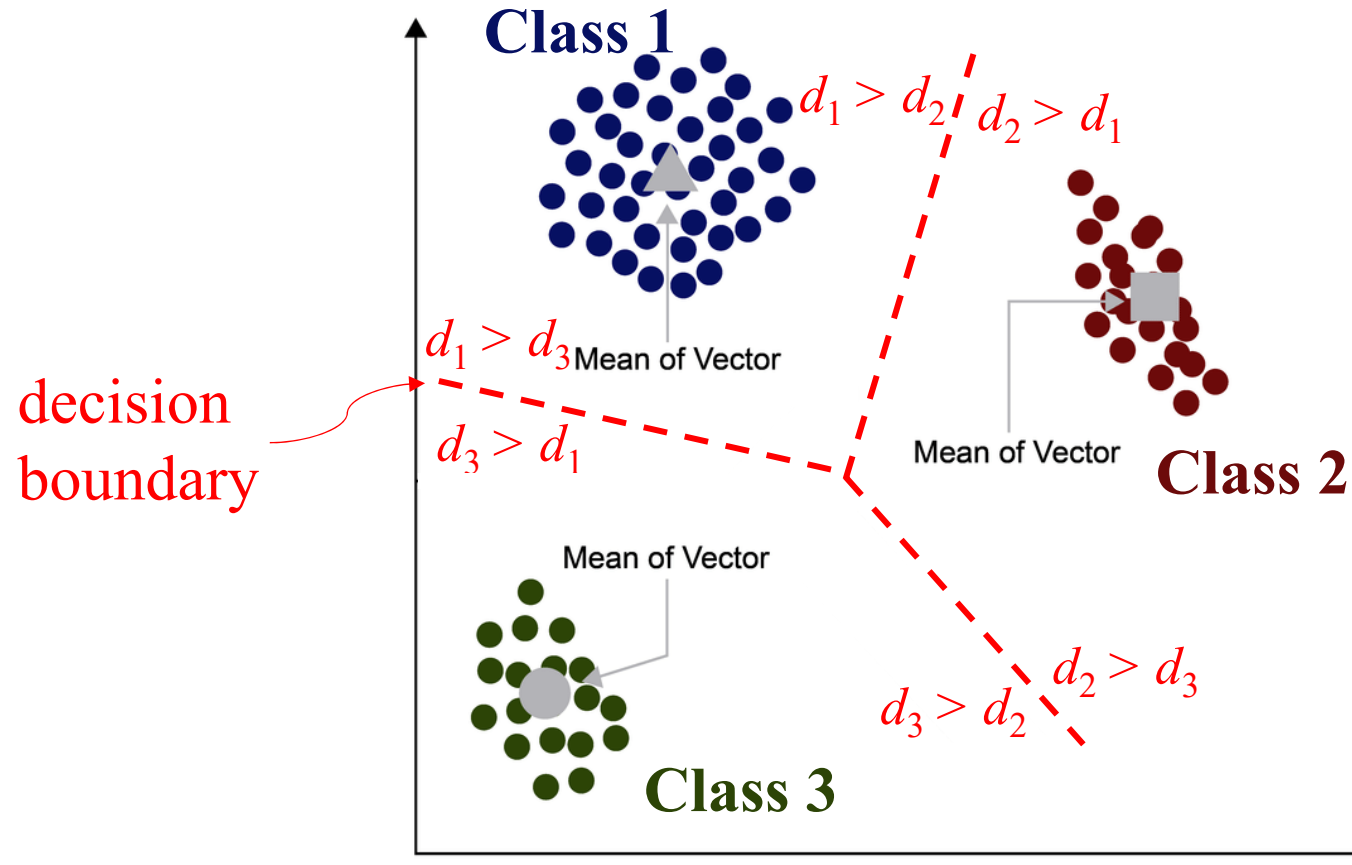
# Minimum Distance Classifier

---

- Let  $d_j = (\mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j)$  . Then, assigning an unknown pattern  $\mathbf{x}$  to the class whose prototype yielded the largest value of  $d$ .
- That is,  $\mathbf{x}$  is assigned to class  $c_i$  , if  $d_i(\mathbf{x}) > d_j(\mathbf{x})$  for  $j = 1, 2, \dots, N_c, j \neq i$ .
- When used for recognition, functions of this form are referred to as *decision* or *discriminant functions*.
- The *decision boundary* separating class  $c_i$  from  $c_j$  is given by the values of  $\mathbf{x}$  for which  $d_i(\mathbf{x}) = d_j(\mathbf{x})$  , or  $d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$ .

# Minimum Distance Classifier

- Decision boundary



# Minimum Distance Classifier

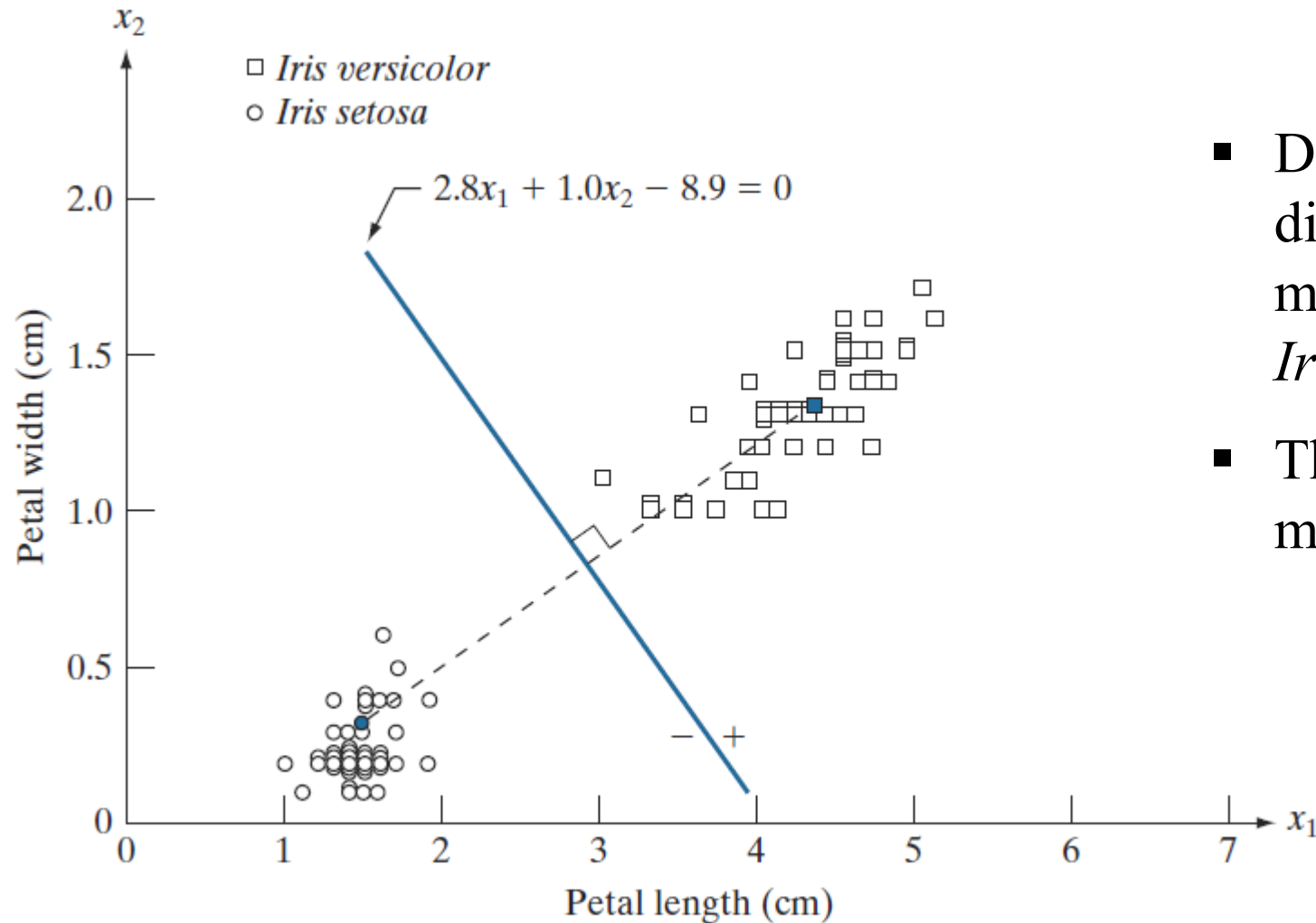
---

- The decision boundary separating class  $c_i$  between  $c_j$  for a minimum distance classifier is:

$$\begin{aligned}d_{ij}(\mathbf{x}) &= d_i(\mathbf{x}) - d_j(\mathbf{x}) \\ &= (\mathbf{m}_i - \mathbf{m}_j)^T \mathbf{x} - \frac{1}{2} (\mathbf{m}_i - \mathbf{m}_j)^T (\mathbf{m}_i - \mathbf{m}_j) = 0\end{aligned}$$

- The boundary given by above equation is the **perpendicular bisector** of the line segment joining  $\mathbf{m}_i$  and  $\mathbf{m}_j$ .

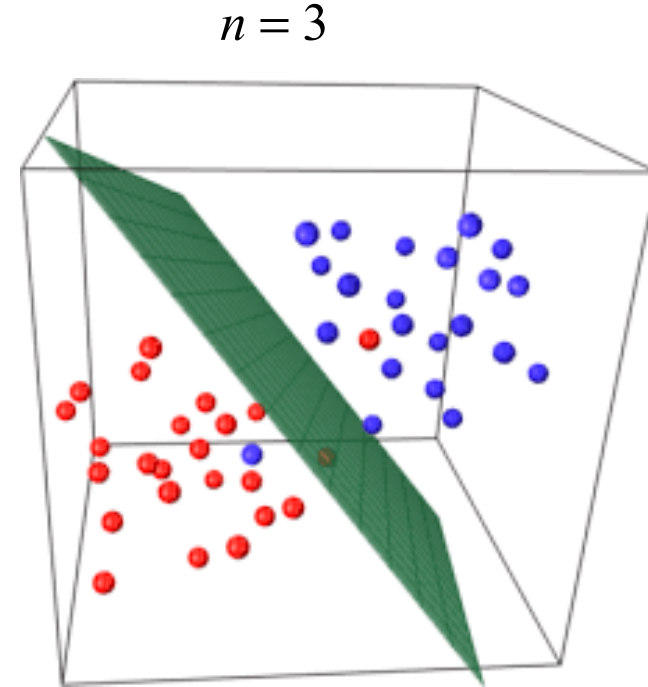
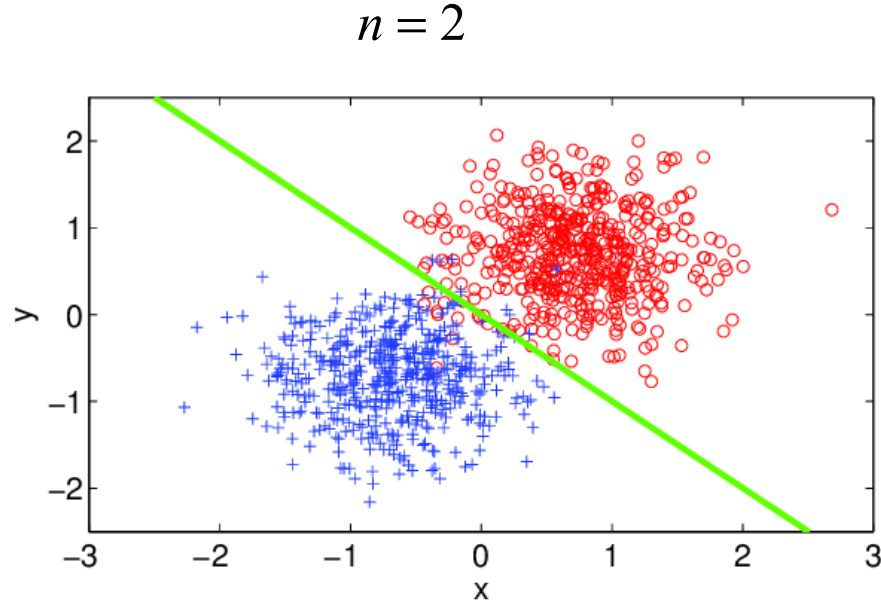
# Minimum Distance Classifier



- Decision boundary of a minimum distance classifier (based on two measurements) for the classes of *Iris versicolor* and *Iris setosa*.
- The dark dot and square are the means of the two classes.

# Minimum Distance Classifier

- In 2-D (i.e.,  $n = 2$ ), the perpendicular bisector is a **line**, for  $n = 3$  it is a **plane**, and for  $n > 3$  it is called a *hyperplane*.

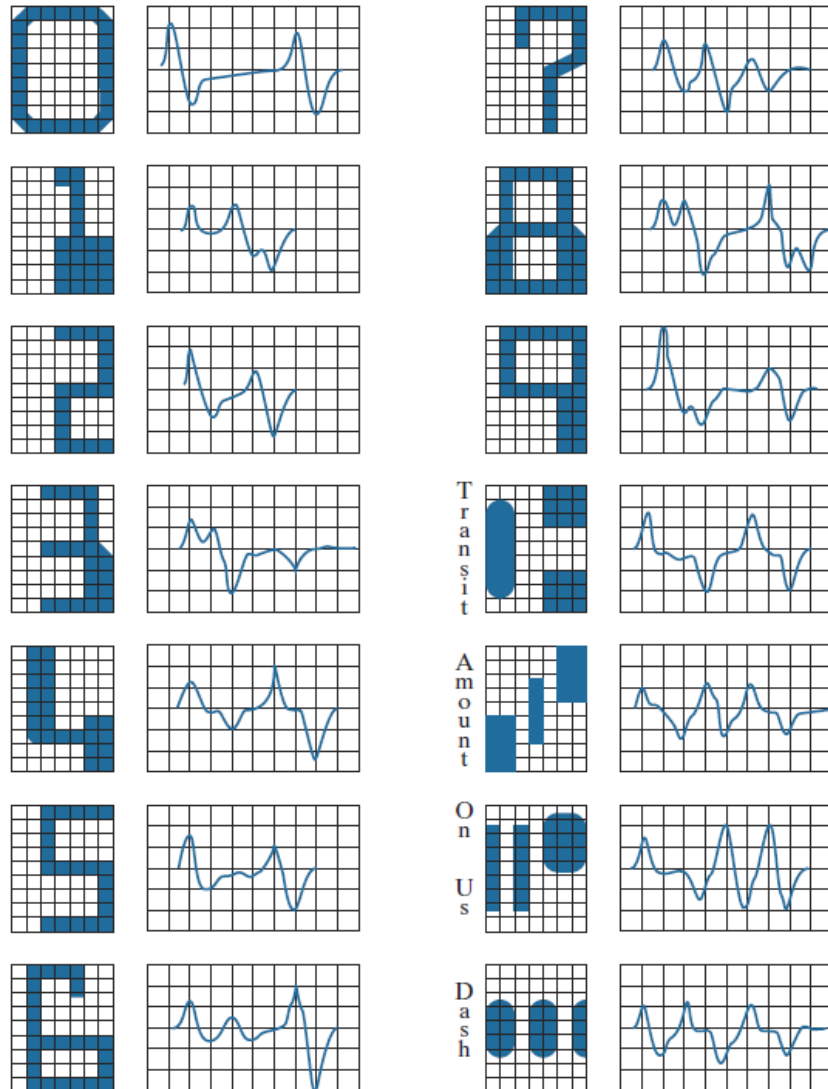


# Minimum Distance Classifier

---

- One of the keys to accurate recognition performance is to specify features that are effective discriminators between classes.
- As a rule, the better the features are at meeting this objective, the better the recognition performance will be.
- In the case of the minimum-distance classifier this implies wide separation between means and tight grouping of the classes.

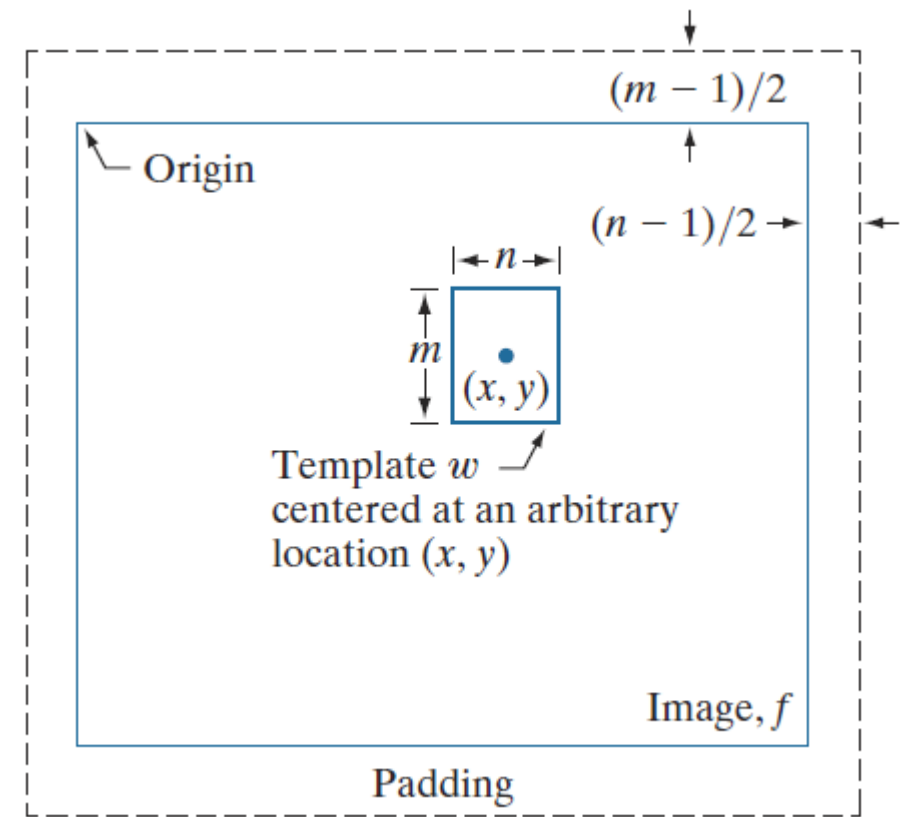
# Minimum Distance Classifier



- Example: Systems based on the Banker's Association E-13B font character are a classic example of how highly engineered features can be used in conjunction with a simple classifier to achieve superior results.
- The characters are stylized to maximize the difference between them.
- The effectiveness of these highly engineered features is further refined by the magnetized ink, which results in clean waveforms with almost no scatter.

# Template Matching

- Template matching: using correlation for 2-D prototype matching
  - Use a template to visit every pixel of an image;
  - Calculate the correlation coefficient between the template and the image in the region shared by them;
  - The pixel with the maximum correlation indicates the location of the best possible match.



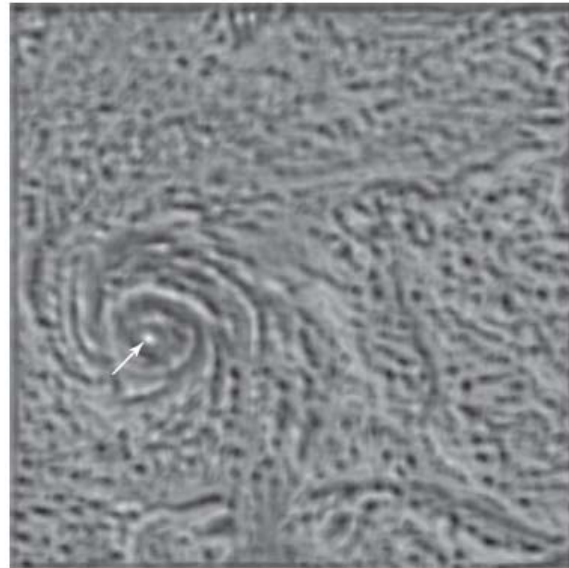



# Template Matching

A  $913 \times 913$  satellite image of a hurricane.



Correlation coefficient shown as an image (note the brightest point, indicated by an arrow).



 A  $31 \times 31$  template of the eye of the storm.

Location of the best match (identified by the arrow). This point is a single pixel, but its size was enlarged to make it easier to see.



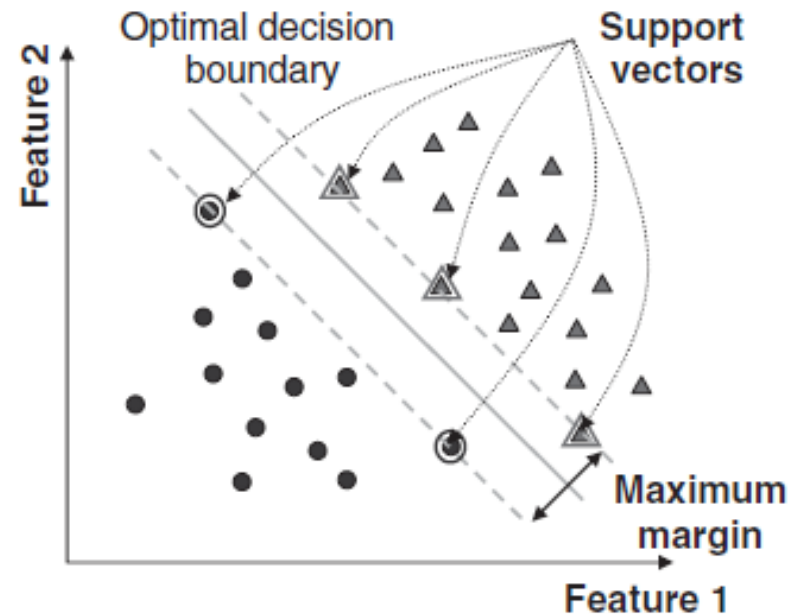
# Matching Structural Prototypes

---

- Other features (such as scale-invariant feature transform) can also be used for matching.
- Image pattern recognition can be achieved by capitalizing precisely on structural relationships inherent in pattern shapes.
- Two basic approaches for the recognition of boundary shapes based on string representations:
  - Matching shape numbers
  - Matching strings

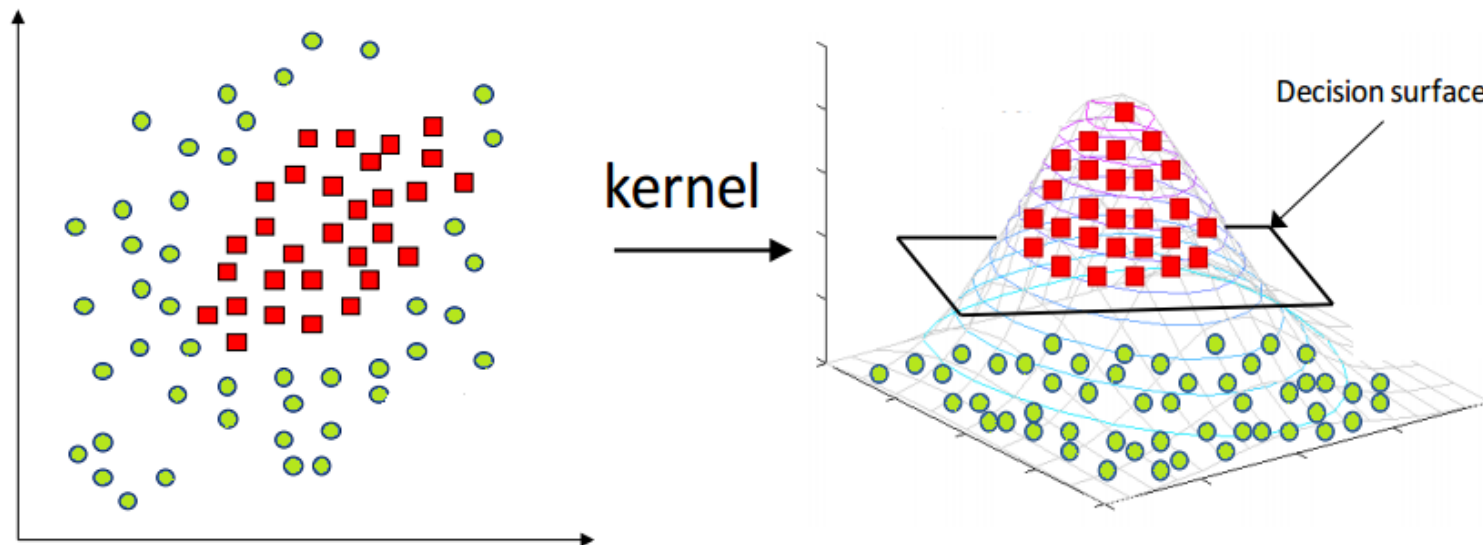
# Support Vector Machine

- **Support Vector Machine (SVM)**: to construct a hyperplane in a high-dimensional space so that an optimal linear decision boundary (which maximizes the margin between the boundaries of different classes) can be achieved in this hyperplane.



# Support Vector Machine

- The novelty of SVM is its ability to use a **kernel function** to map the original feature space (*where classes are not linearly separable*) into a higher dimensional space (*where the classes are linearly separable*) and find the support vectors in that higher dimensional space.



# Optimal (Bayes) Classifiers

---

- **Bayes Classifiers:** A probabilistic approach to pattern classification.
- As is true in most fields that deal with measuring and interpreting physical events, probability considerations become important in pattern recognition because of the *randomness* under which pattern classes normally are generated.
- It is possible to derive a classification approach that is optimal in the sense that, on average, it yields the *lowest probability of committing classification errors*.

# Optimal (Bayes) Classifiers

---

- Bayes Theorem:

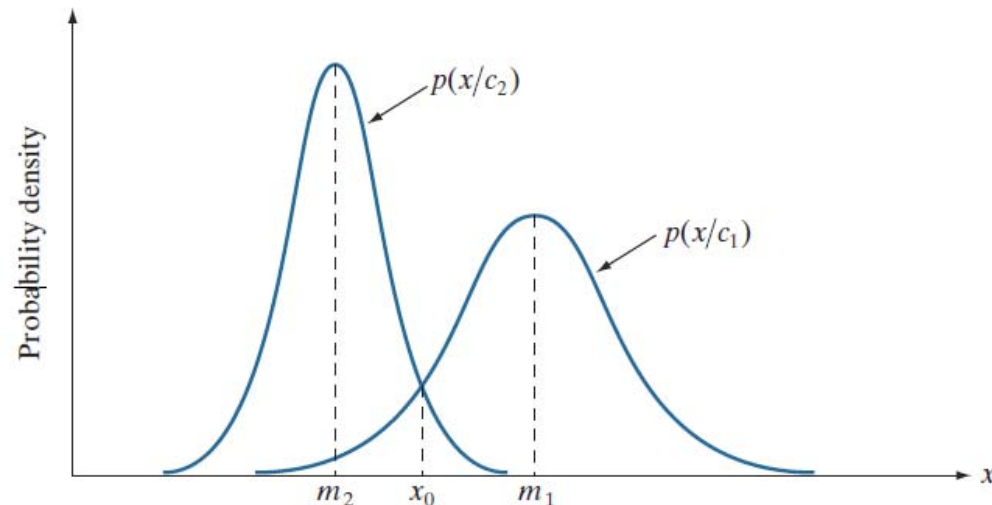
$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

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

- **Posterior**  $P(c_i | \mathbf{x})$ : the conditional probability of correct class being  $c_i$ , given that feature  $\mathbf{x}$  has been observed;
- **Likelihood**  $p(\mathbf{x} | c_i)$ : the conditional probability that a feature vector  $\mathbf{x}$  belongs to a class  $c_i$ ;
- **Prior**  $P(c_i)$ : the previously known probability of correct class being class  $c_i$  (without regarding the observed feature);
- **Evidence**  $p(\mathbf{x})$ : the total probability of observing the feature vector  $\mathbf{x}$ .

# Optimal (Bayes) Classifiers

- The posterior probability is calculated for each class, given  $\mathbf{x}$ , and then the final classification assigns  $\mathbf{x}$  to the class for which the posterior probability is the largest.
- It can be shown that the Bayes classifiers are optimal in the sense that they minimize the average loss in misclassification.



- Probability density functions for two 1-D pattern classes.
- Point  $x_0$  (at the intersection of the two curves) is the Bayes decision boundary if the two classes are equally likely to occur.

# Optimal (Bayes) Classifiers

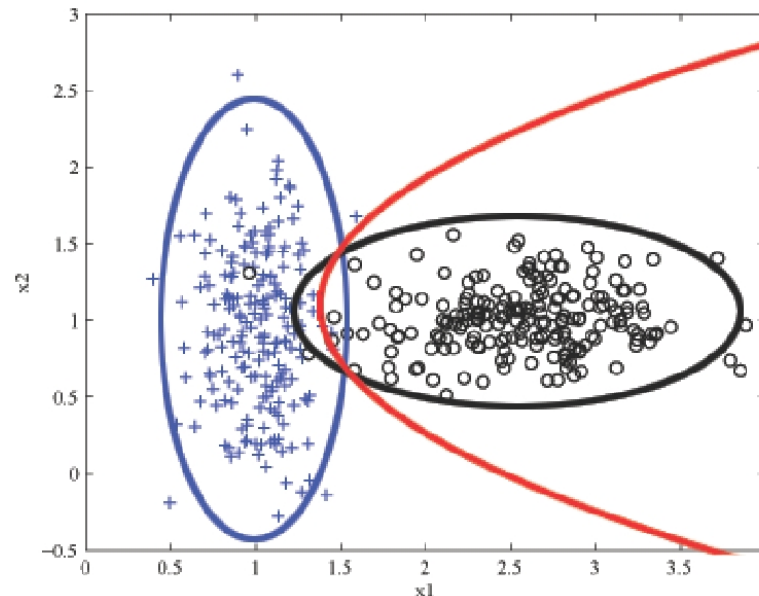
---

- In order to build a classifier, these probability distributions must either be known ahead of time or must be learned from the data.
- But it is difficult to estimate these probabilities, particularly for high dimensional features.
- A **Naïve Bayes** classifier estimates a univariate Gaussian distribution for each class and each feature based on the assumption that the presence or absence of a particular feature is unrelated to any other features, given the class labels.



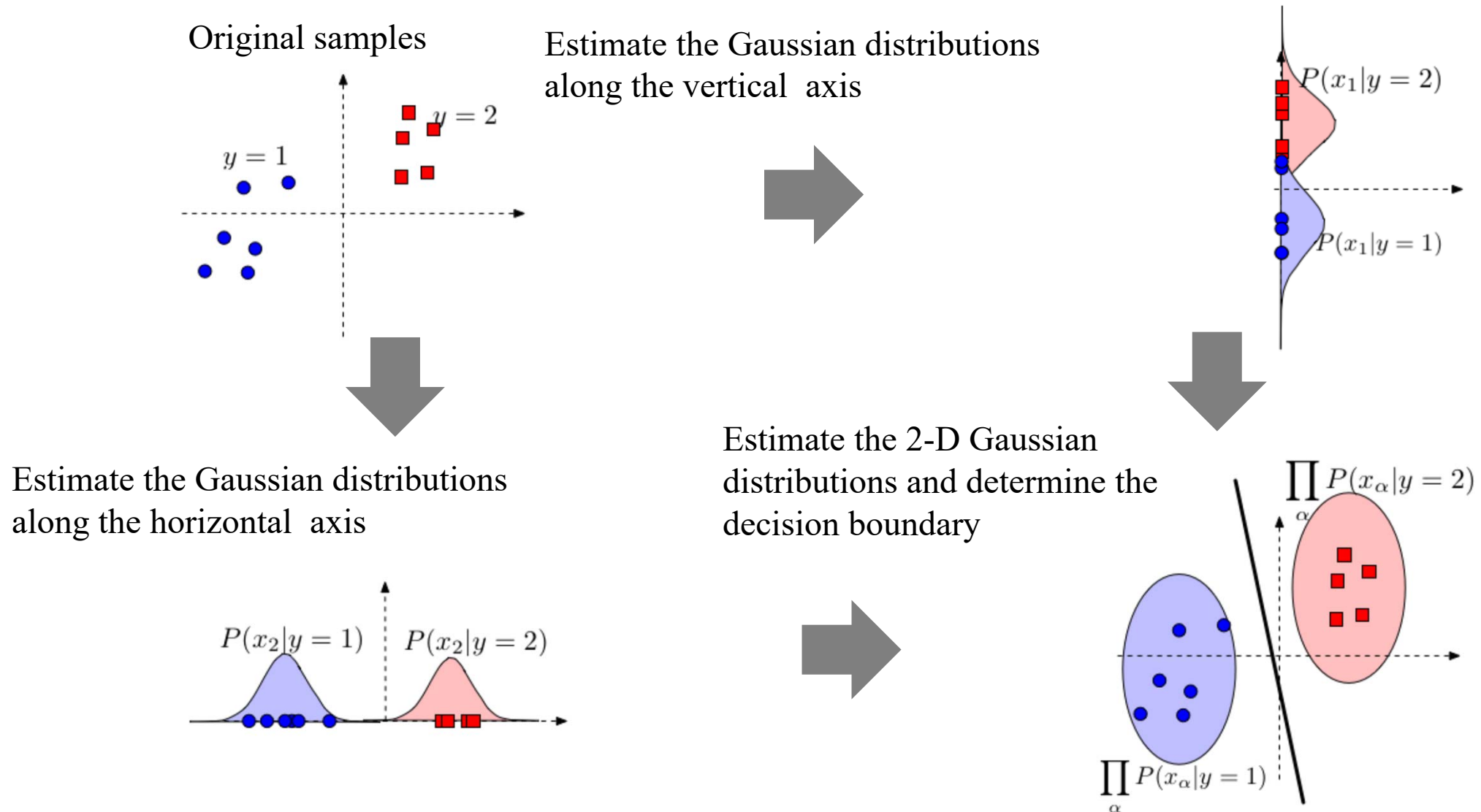
# Optimal (Bayes) Classifiers

- In practice, Naïve Bayes has been shown to provide good performance, even under mild violations of the independence assumptions.
- Naïve Bayes only requires a small amount of training data to estimate the Gaussian parameters necessary for classification.



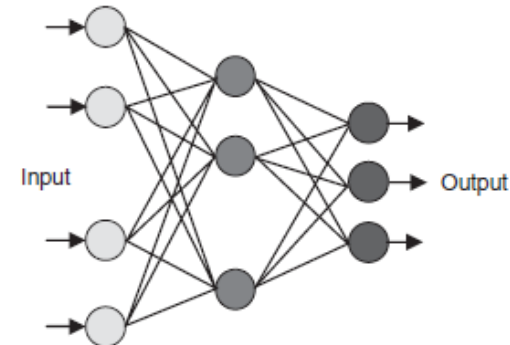
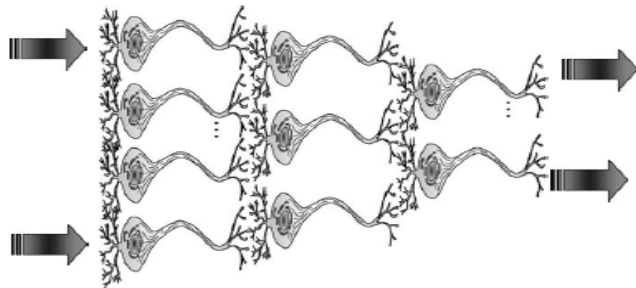
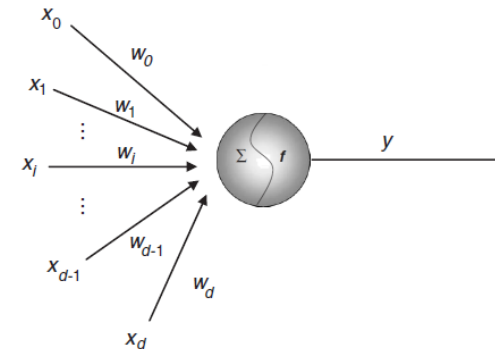
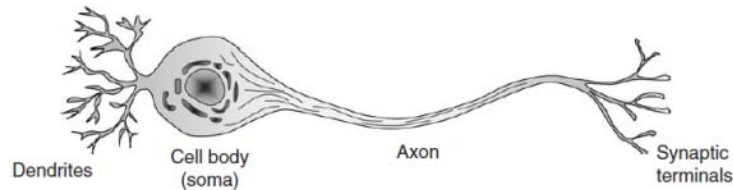
2D binary classification with Naïve Bayes.  
A density contour is drawn for the Gaussian model of each class and the decision boundary is shown in red.

# Optimal (Bayes) Classifiers



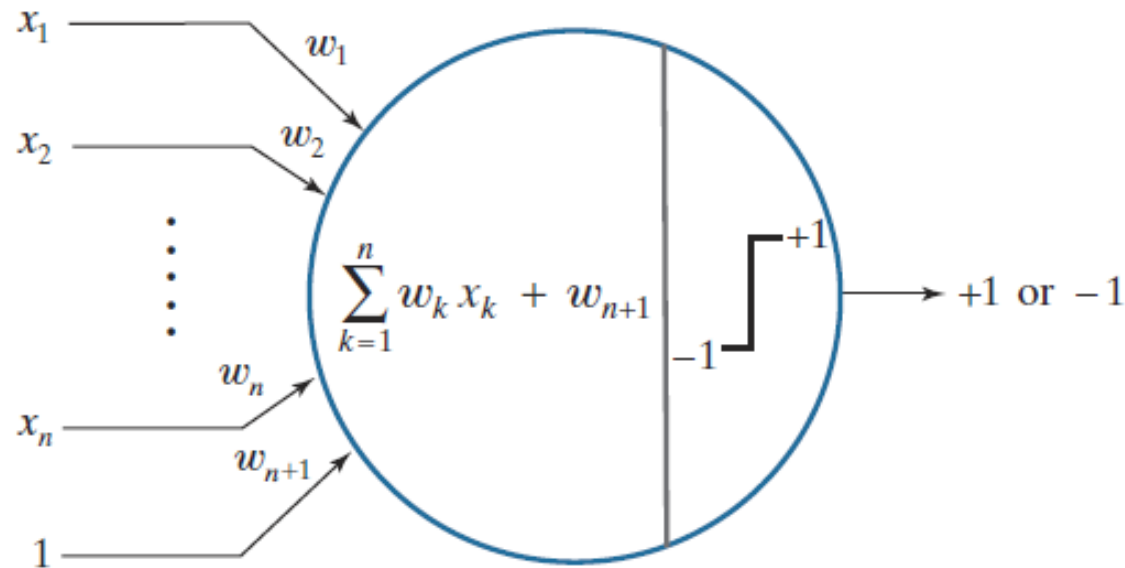
# Neural Networks

- **Artificial Neural Network (ANN):** an information processing paradigm that is inspired by the way the brain processes information
- It is composed of a large number of highly interconnected nodes working together to solve specific problems.



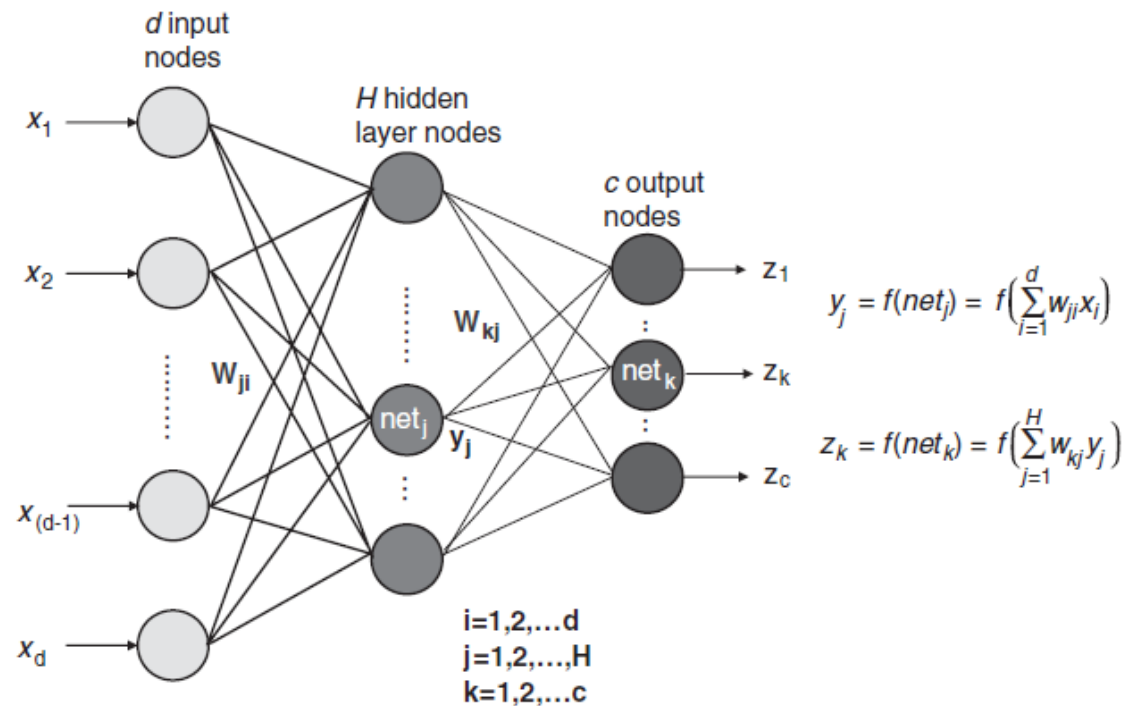
# Neural Networks

- A neuron is modeled by a node.
- The output of the node is calculated as the weighted sum of its inputs, called the *net sum*, or simply *net*, passed through an activation function  $f$ .



# Neural Networks

- **Multilayer Perceptron (MLP)**: to mimic the structure of the nervous system.
- A large number of neurons are interconnected; the information flows from one neuron to others in a cascade-like structure until it reaches its intended destination.



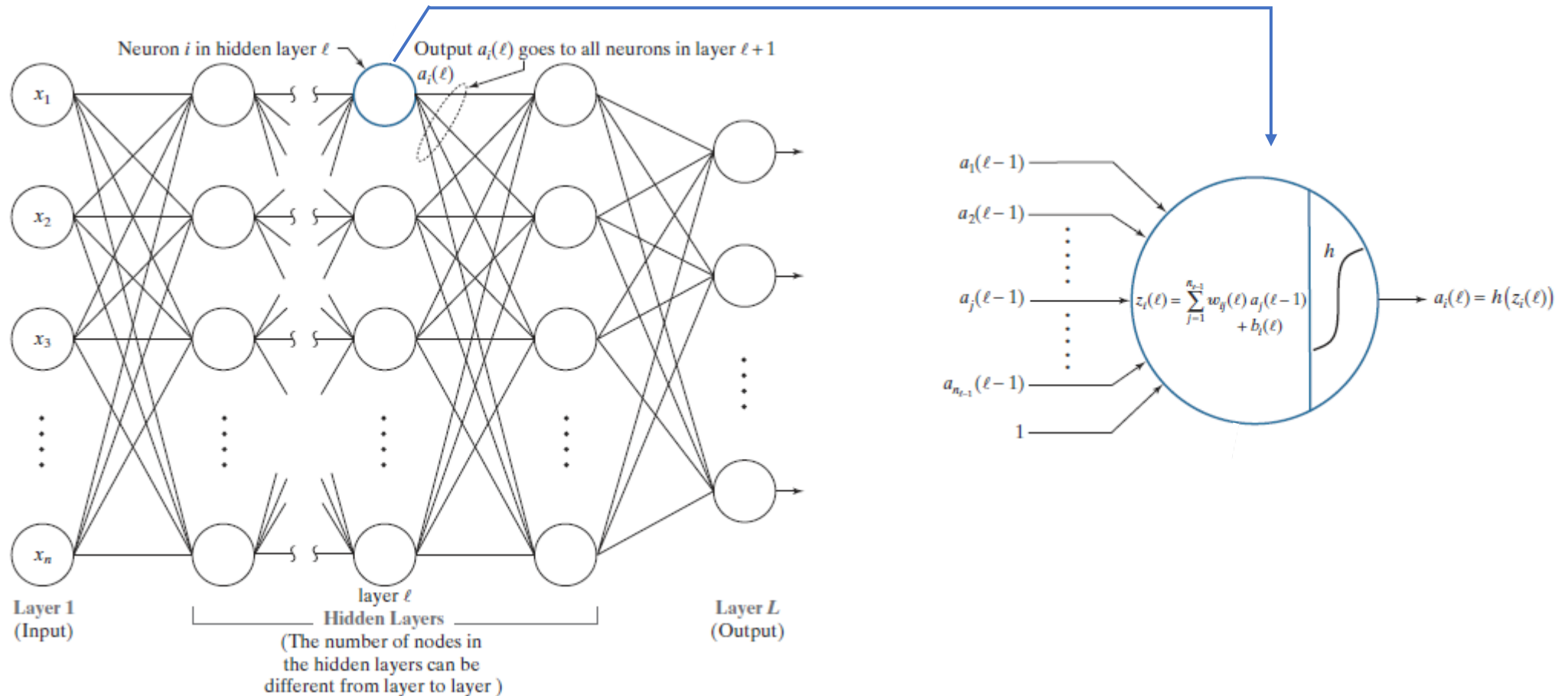
# Deep Learning

---

- A two-layer ANN with a finite number of hidden units can approximate any continuous nonlinear function, and thus it is regarded as universal approximator.
- It is also possible to better approximate complex functions using a ‘**deep**’ architecture, i.e., more than two layers or more hidden layers.
- **Deep learning** is a class of machine learning algorithms that uses a cascade of many layers of nonlinear processing units for feature extraction and transformation.

# Deep Learning

- General model of a feedforward, fully connected neural net.



# Convolutional Neural Networks

---

- Up to this point, we have organized pattern features as vectors. Generally, this assumes that the form of those features has been specified (i.e., “engineered”) and extracted from images prior to being input to a neural network.
- But one of the strengths of neural networks is that they are capable of learning pattern features directly from training data.
- What we would like to do is input a set of training images **directly** into a neural network, and have the network learn the necessary features *on its own*.
- One way to do this would be to convert images to vectors directly by organizing the pixels based on a linear index, and then letting each element (pixel) of the linear index be an element of the vector.



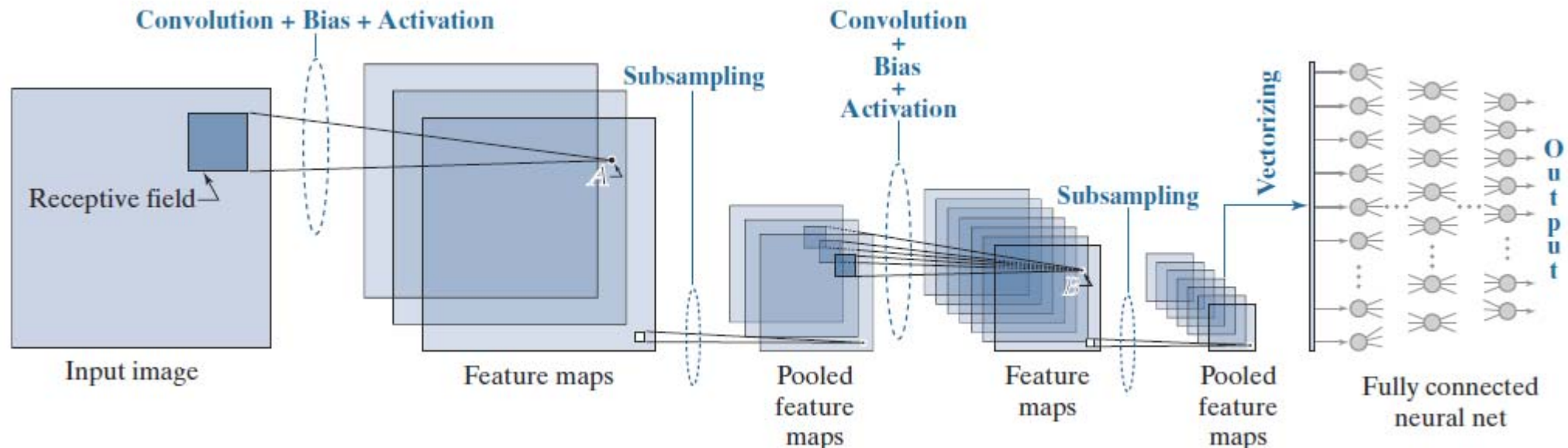
# Convolutional Neural Networks

---

- However, this approach (input vectorized image) does not utilize any spatial relationships that may exist between pixels in an image, such as pixel arrangements into corners, the presence of edge segments, and other features that may help to differentiate one image from another.
- A class of neural networks called **convolutional neural networks (CNNs)** can accept images as inputs and are ideally suited for automatic learning and image classification (end-to-end).

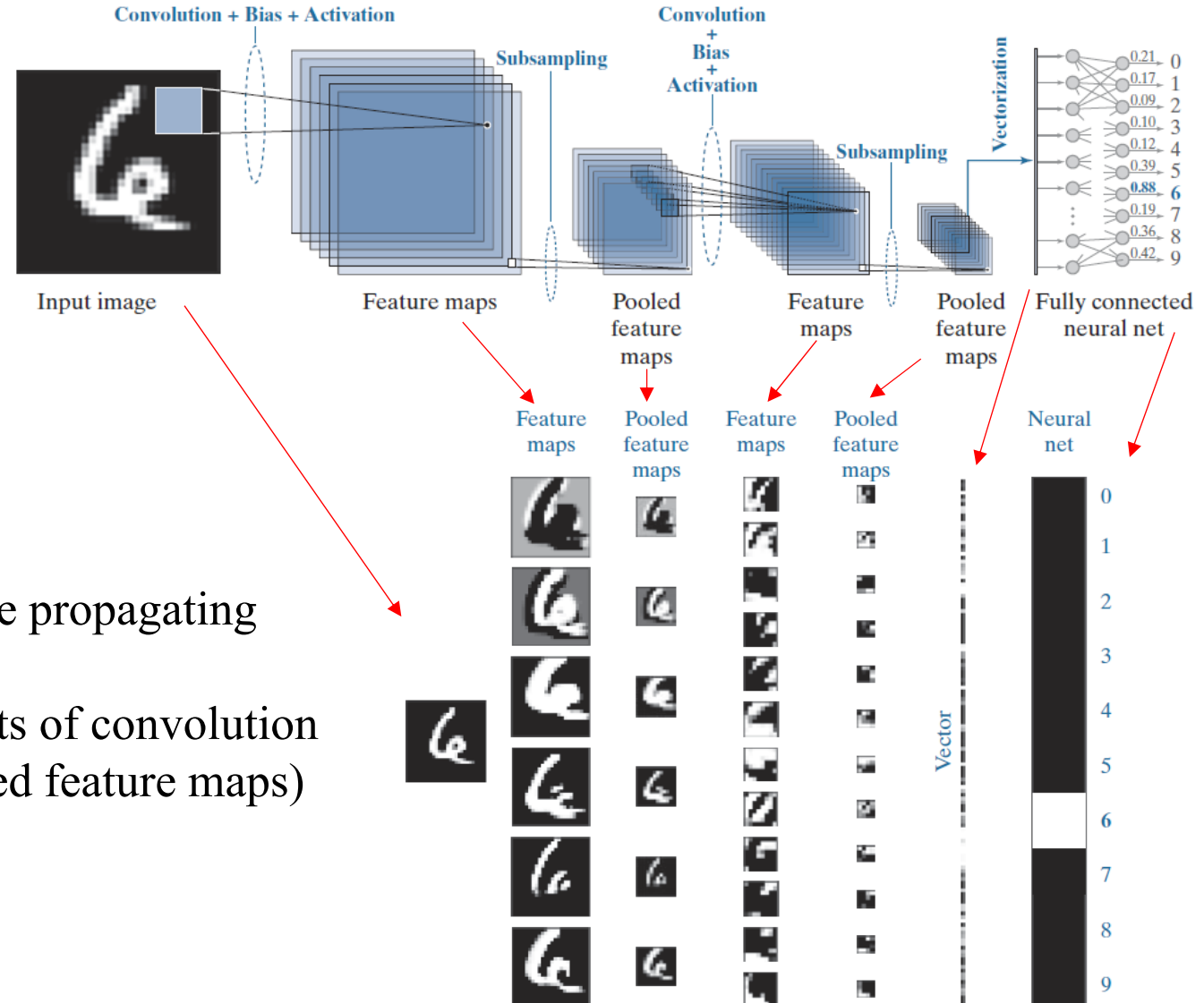
# Convolutional Neural Networks

- A CNN architecture.
  - The convolutional layer applies filters to the input image to extract features, the pooling layer downsamples the image to reduce computation
  - The last pooled feature maps are vectorized and serve as the input to a fully connected neural network.
  - The class to which the input image belongs is determined by the output neuron with the highest value.



# Convolutional Neural Networks

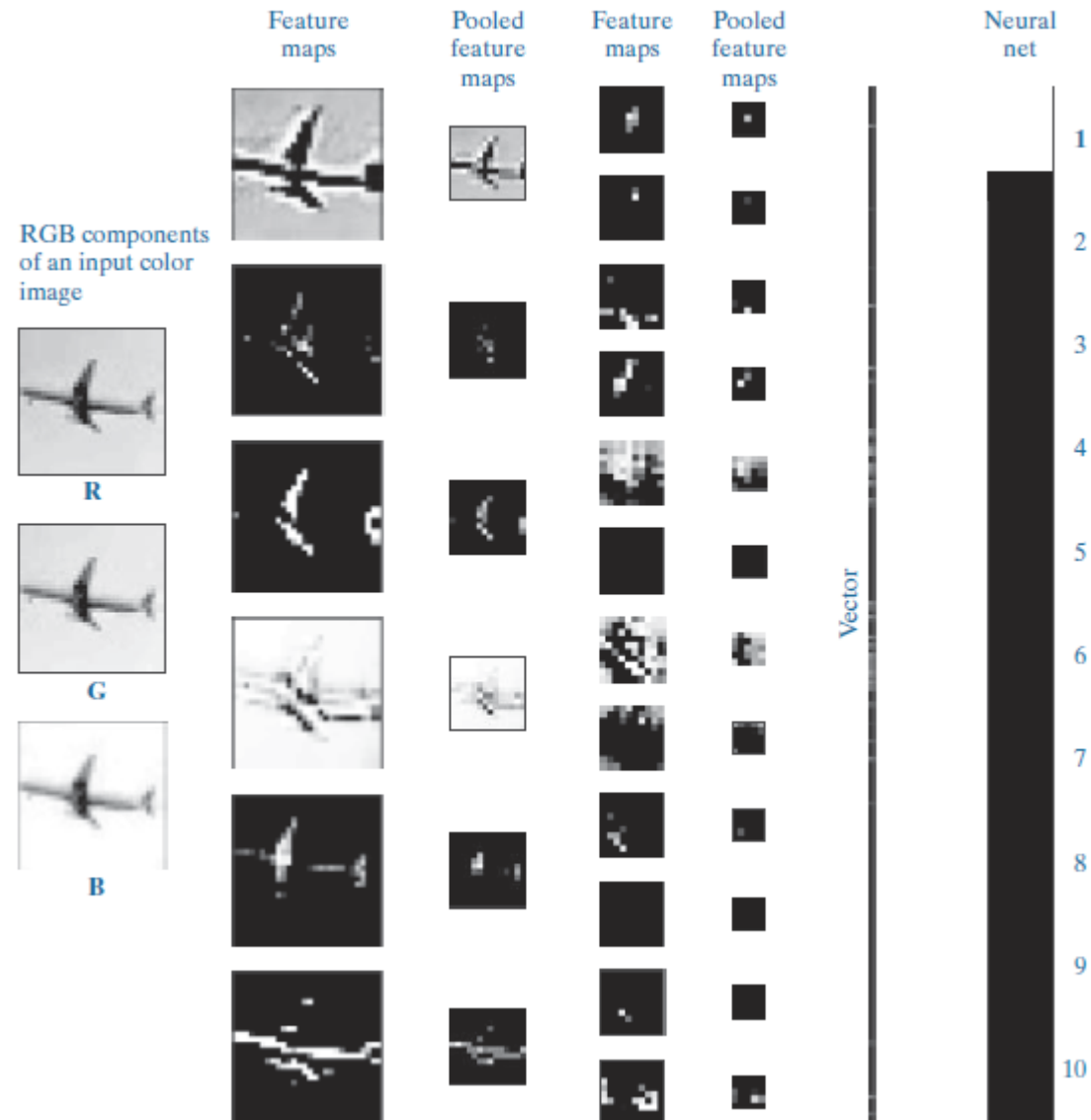
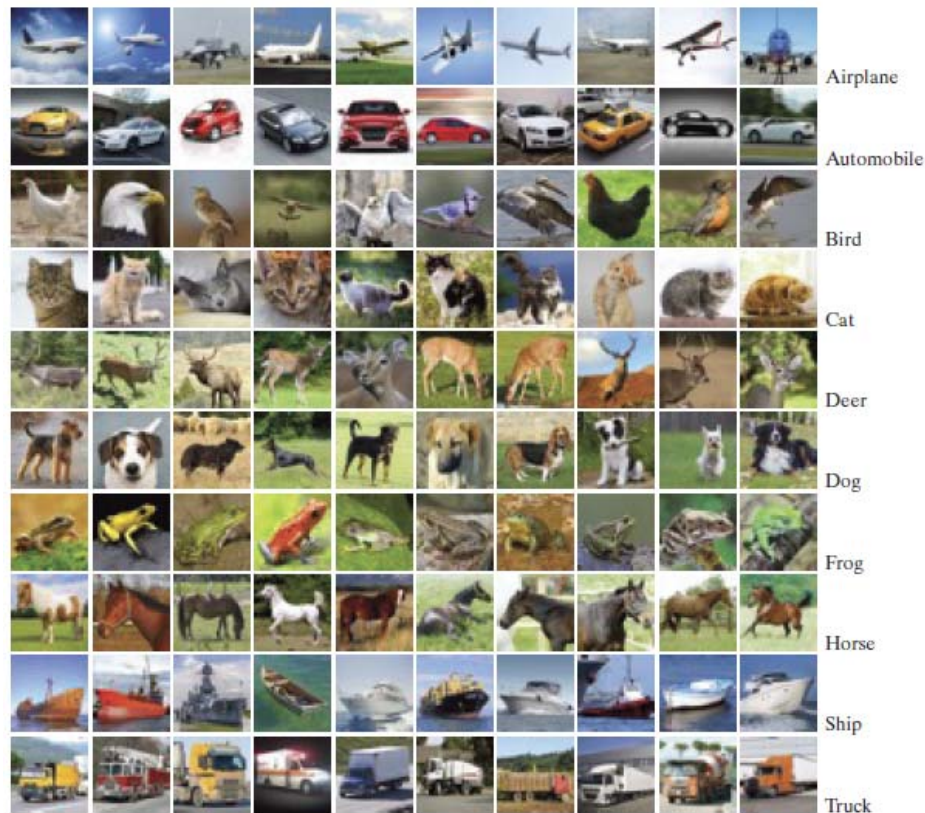
Numerical example illustrating the various functions of a CNN, including recognition of an input image.



- Visual summary of an input image propagating through the CNN.
- Shown as images are all the results of convolution (feature maps) and pooling (pooled feature maps) for both layers of the network.

# Pattern Classification

Mini images of size  $32 \times 32$  pixels,  
representative of the 50,000 training and  
10,000 test images in the CIFAR-10 database



Graphical  
illustration of  
a forward pass  
through the  
trained CNN

# Summary

---

- In this lecture we have learnt:
  - Background of Pattern Classification
  - Patterns and Pattern Classes
  - Pattern Classification by Prototype Matching
  - Optimal (Bayes) Statistical Classifiers
  - Neural Networks and Deep Learning
  - Convolutional Neural Networks

# Optional Homework

---

Check the Textbook!

- **Chapter 12: Problems 12.1, 12.9 , 12.17, 12.23, 12.30**
- Homework answers will be provided at the end of each week.