Introduction to Pattern Recognition
Part II

Selim Aksoy
Department of Computer Engineering
Bilkent University
saksoy@cs.bilkent.edu.tr

Overview

- Statistical pattern recognition
  - Bayesian Decision Theory
    - Parametric models
    - Non-parametric models
  - Feature reduction and selection
  - Non-Bayesian classifiers
    - Distance-based classifiers
    - Decision boundary-based classifiers
  - Unsupervised learning and clustering
  - Algorithm-independent learning issues
    - Estimating and comparing classifiers
    - Combining classifiers
- Structural and syntactic pattern recognition

Bayesian Decision Theory

- Bayesian Decision Theory is a statistical approach that quantifies the tradeoffs between various decisions using probabilities and costs that accompany such decisions.
- Fish sorting example: define w, the type of fish we observe (state of nature), as a random variable where
  - $w = w_1$ for sea bass,
  - $w = w_2$ for salmon.
  - $P(w_1)$ is the *a priori probability* that the next fish is a sea bass.
  - $P(w_2)$ is the a priori probability that the next fish is a salmon.

Prior Probabilities

- Prior probabilities reflect our knowledge of how likely each type of fish will appear before we actually see it.
- How can we choose $P(w_1)$ and $P(w_2)$?
  - Set $P(w_1) = P(w_2)$ if they are equiprobable (*uniform priors*).
  - May use different values depending on the fishing area, time of the year, etc.
- Assume there are no other types of fish

$$P(w_1) + P(w_2) = 1$$

(exclusivity and exhaustivity).

Making a Decision

- How can we make a decision with only the prior information?

$$\text{Decide} \quad \begin{cases} w_1 & \text{if } P(w_1) > P(w_2) \\ w_2 & \text{otherwise} \end{cases}$$

- What is the *probability of error* for this decision?

$$P(error) = \min\{P(w_1), P(w_2)\}$$

Class-conditional Probabilities

- Let's try to improve the decision using the lightness measurement x.
- Let x be a continuous random variable.
- Define $p(x|w_j)$ as the *class-conditional probability density* (probability of x given that the state of nature is w_j for $j = 1, 2$).
- $p(x|w_1)$ and $p(x|w_2)$ describe the difference in lightness between populations of sea bass and salmon.

Posterior Probabilities

- Suppose we know $P(w_j)$ and $p(x|w_j)$ for $j = 1, 2$, and measure the lightness of a fish as the value x.
- Define $P(w_j|x)$ as the *a posteriori probability* (probability of the state of nature being w_j given the measurement of feature value x).
- We can use the *Bayes formula* to convert the prior probability to the posterior probability

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

where $p(x) = \sum_{j=1}^{2} p(x|w_j)P(w_j)$.

Making a Decision

- $p(x|w_j)$ is called the *likelihood* and $p(x)$ is called the *evidence*.
- How can we make a decision after observing the value of x?

$$\text{Decide} \quad \begin{cases} w_1 & \text{if } P(w_1|x) > P(w_2|x) \\ w_2 & \text{otherwise} \end{cases}$$

- Rewriting the rule gives

$$\text{Decide} \quad \begin{cases} w_1 & \text{if } \frac{p(x|w_1)}{p(x|w_2)} > \frac{P(w_2)}{P(w_1)} \\ w_2 & \text{otherwise} \end{cases}$$

- Note that, at every x, $P(w_1|x) + P(w_2|x) = 1$.

Making a Decision

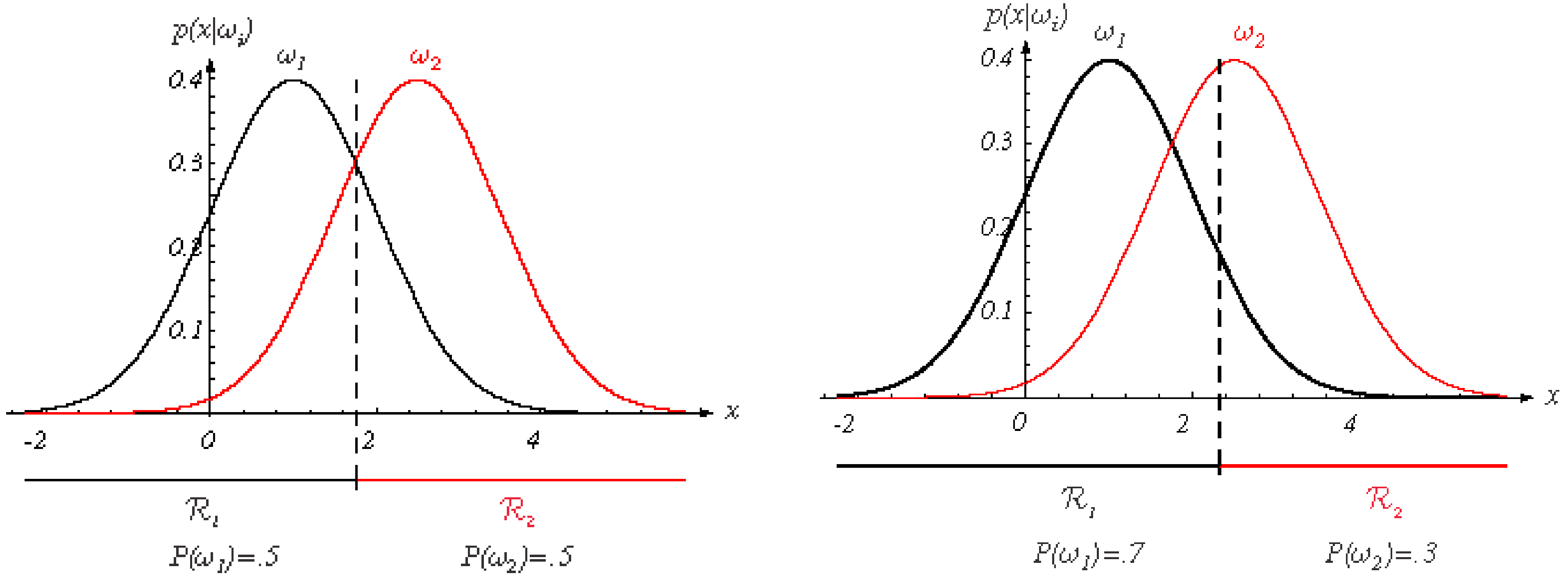


Figure 1: Optimum thresholds for different priors.

Probability of Error

- What is the probability of error for this decision?

$$P(error|x) = \begin{cases} P(w_1|x) & \text{if we decide } w_2 \\ P(w_2|x) & \text{if we decide } w_1 \end{cases}$$

- What is the average probability of error?

$$P(error) = \int_{-\infty}^{\infty} p(error, x)\, dx = \int_{-\infty}^{\infty} P(error|x)\, p(x)\, dx$$

- *Bayes decision rule* minimizes this error because

$$P(error|x) = \min\{P(w_1|x), P(w_2|x)\}.$$

Probability of Error

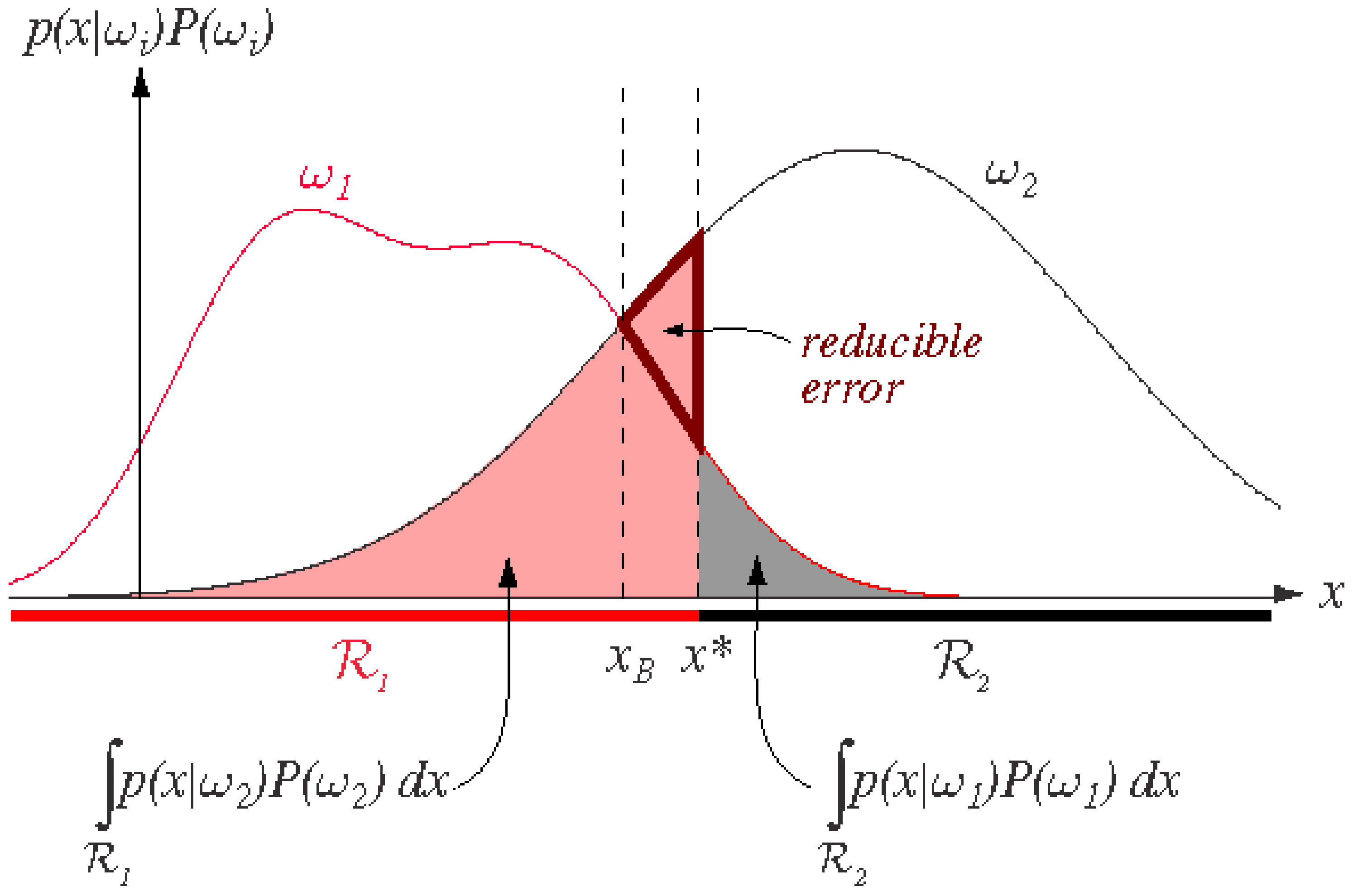


Figure 2: Components of the probability of error for equal priors and the non-optimal decision point x^*. The optimal point x_B minimizes the total shaded area and gives the Bayes error rate.

Receiver Operating Characteristics

- Consider the two-category case and define
  - w_1: target is present,
  - w_2: target is not present.

Table 1: *Confusion matrix*.

		Assigned	
		w_1	w_2
True	w_1	correct detection	mis-detection
	w_2	false alarm	correct rejection

- Mis-detection is also called false negative or Type I error.
- False alarm is also called false positive or Type II error.

Receiver Operating Characteristics

- If we use a parameter (e.g., a threshold) in our decision, the plot of these rates for different values of the parameter is called the *receiver operating characteristic* (ROC) curve.

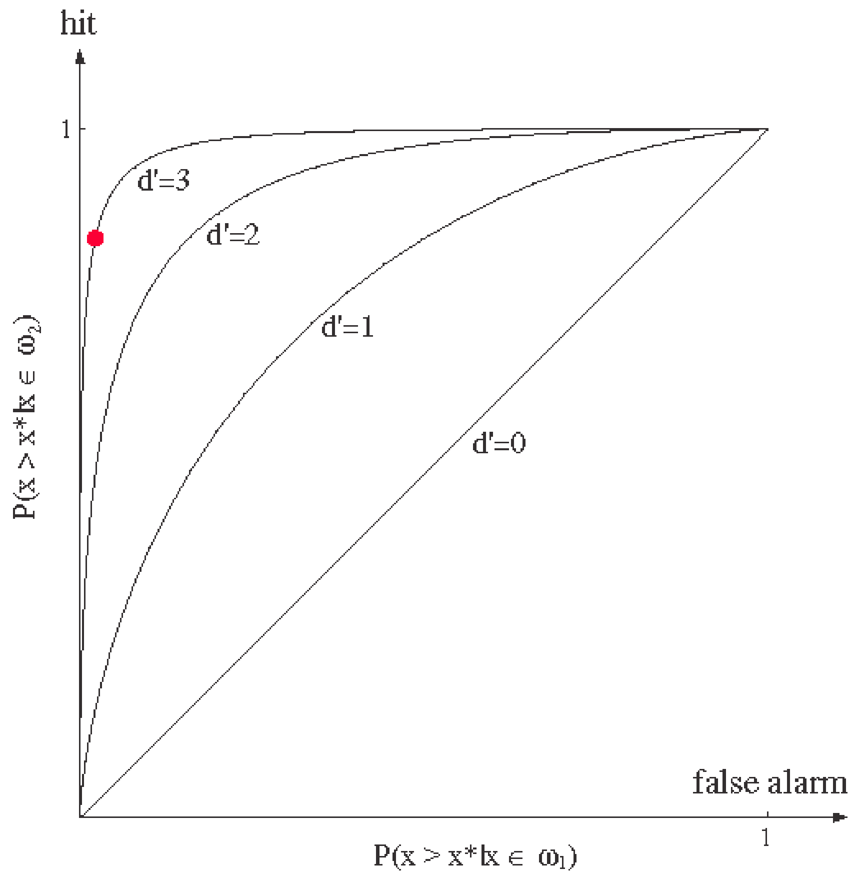


Figure 3: Example receiver operating characteristic (ROC) curves for different settings of the system.

Bayesian Decision Theory

- How can we generalize to
  - ▸ more than one feature?
    - – replace the scalar x by the feature vector $\mathbf{x}$
  - ▸ more than two states of nature?
    - – just a difference in notation
  - ▸ allowing actions other than just decisions?
    - – allow the possibility of rejection
  - ▸ different risks in the decision?
    - – define how costly each action is

Minimum-error-rate Classification

- Let $\{w_1, \ldots, w_c\}$ be the finite set of c states of nature (*classes*, *categories*).
- Let $\mathbf{x}$ be the d-component vector-valued random variable called the *feature vector*.
- If all errors are equally costly, the *minimum-error decision rule* is defined as

$$\text{Decide } w_i \text{ if } P(w_i|\mathbf{x}) > P(w_j|\mathbf{x}) \quad \forall j \neq i.$$

- The resulting error is called the *Bayes error* and is the best performance that can be achieved.

Bayesian Decision Theory

- Bayesian Decision Theory shows us how to design an optimal classifier if we know the prior probabilities $P(w_i)$ and the class-conditional densities $p(\mathbf{x}|w_i)$.
- Unfortunately, we rarely have complete knowledge of the probabilistic structure.
- However, we can often find design samples or *training data* that include particular representatives of the patterns we want to classify.

Bayesian Decision Theory

- How can we estimate (learn) the unknown $p(\mathbf{x}|w_j), j = 1, \ldots, c$?
- Parametric models: assume that the form of the density functions are known.
  - Density models (e.g., Gaussian)
  - Mixture models (e.g., mixture of Gaussians)
  - Hidden Markov Models
  - Bayesian Belief Networks
- Non-parametric models: no assumption about the form.
  - Histogram-based estimation
  - Parzen window estimation
  - Nearest neighbor estimation

The Gaussian Density

- Gaussian can be considered as a model where the feature vectors for a given class are continuous-valued, randomly corrupted versions of a single typical or prototype vector.
- Some properties of the Gaussian:
  - Analytically tractable.
  - Completely specified by the 1st and 2nd moments.
  - Has the maximum entropy of all distributions with a given mean and variance.
  - Many processes are asymptotically Gaussian (Central Limit Theorem).
  - Linear transformations of a Gaussian are also Gaussian.
  - Uncorrelatedness implies independence.

Univariate Gaussian

- For $x \in \mathbb{R}$:

$$\begin{aligned} p(x) &= N(\mu, \sigma^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right] \end{aligned}$$

where

$$\mu = E[x] = \int_{-\infty}^{\infty} x \, p(x) \, dx,$$

$$\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 \, p(x) \, dx.$$

Univariate Gaussian

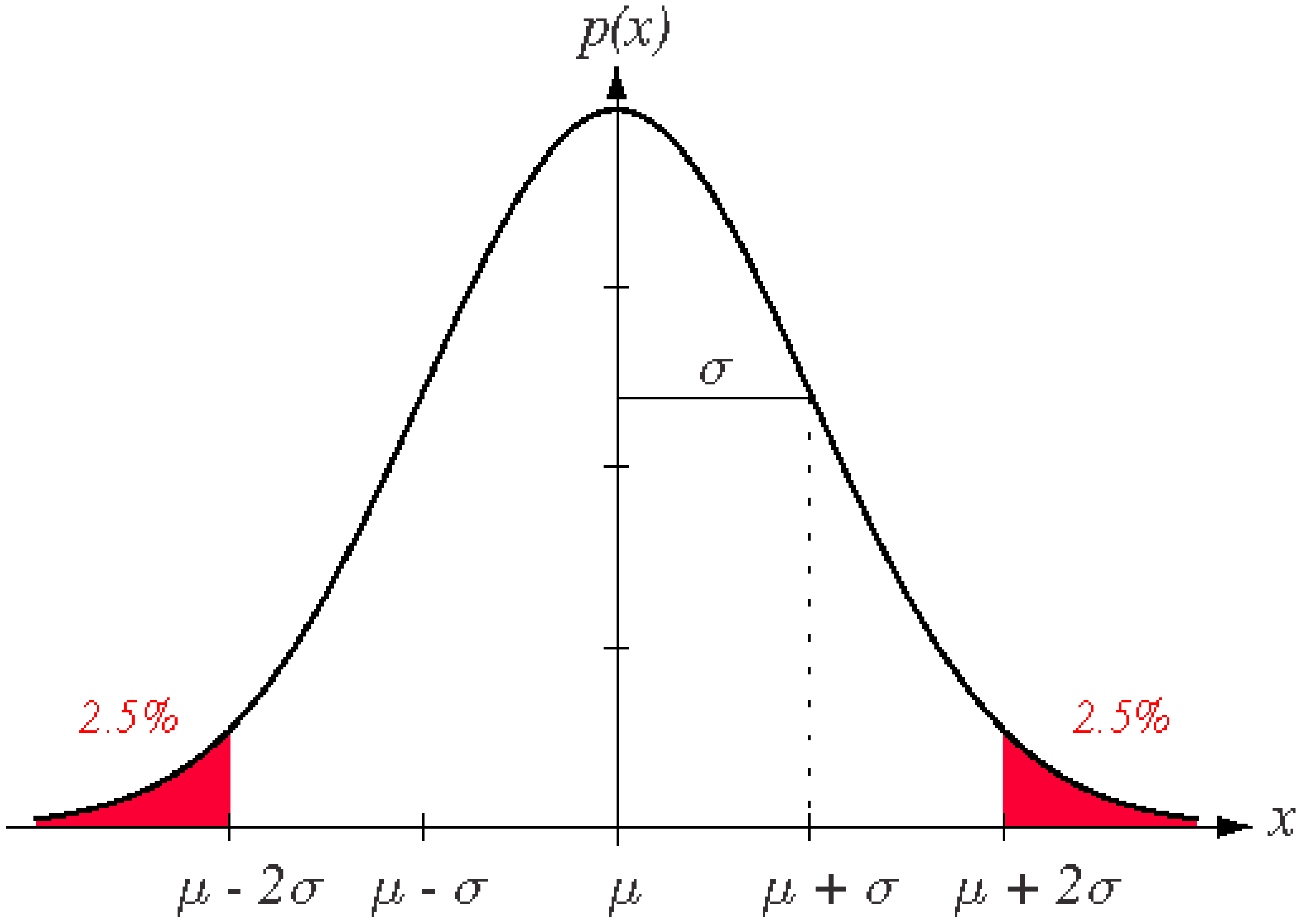


Figure 4: A univariate Gaussian distribution has roughly 95% of its area in the range $|x - \mu| \leq 2\sigma$.

Multivariate Gaussian

- For $\mathbf{x} \in \mathbb{R}^d$:

$$p(\mathbf{x}) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$= \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

where

$$\boldsymbol{\mu} = E[\mathbf{x}] = \int \mathbf{x}\, p(\mathbf{x})\, d\mathbf{x},$$

$$\boldsymbol{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T] = \int (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T p(\mathbf{x})\, d\mathbf{x}.$$

Multivariate Gaussian

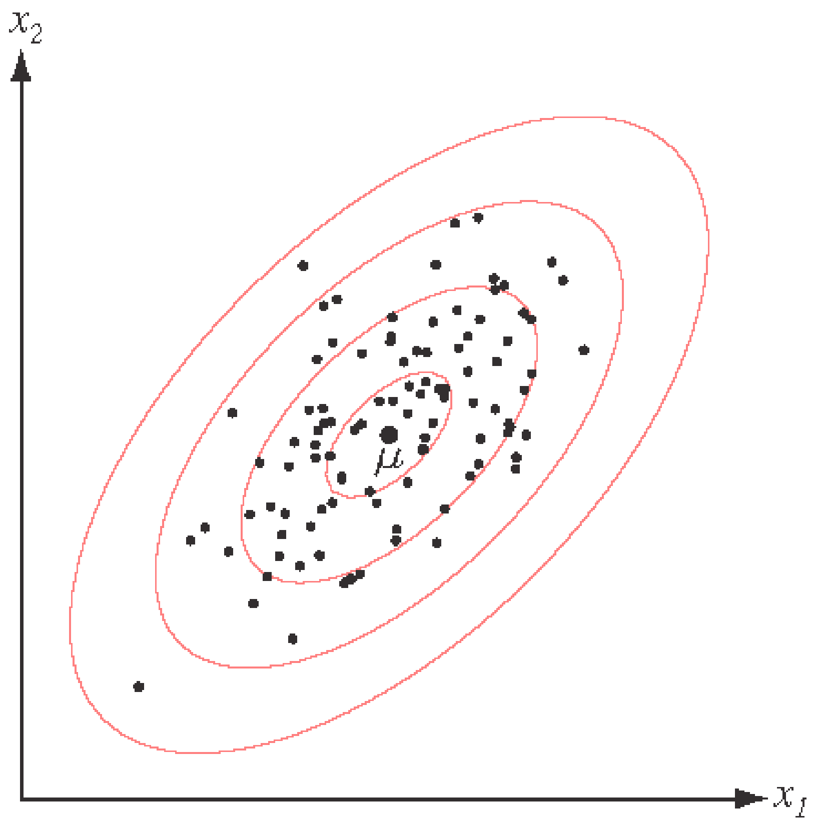
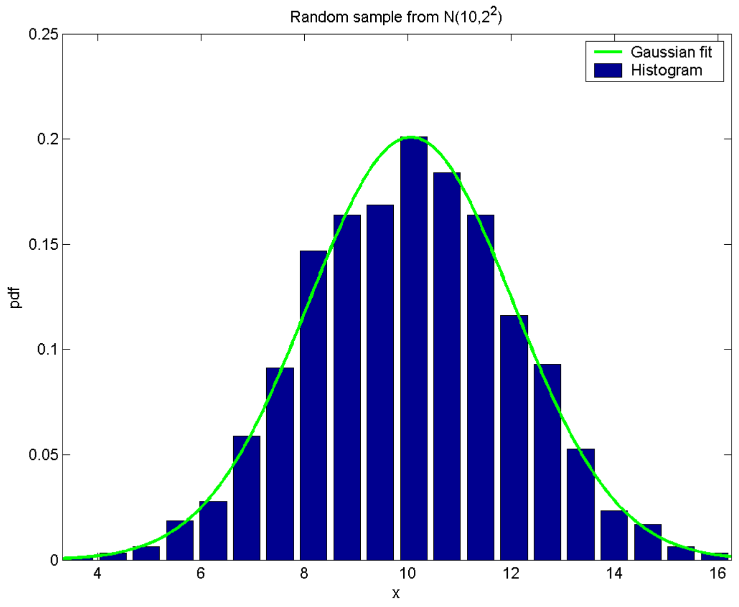


Figure 5: Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean $\boldsymbol{\mu}$. The loci of points of constant density are the ellipses for which $(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is constant, where the eigenvectors of $\boldsymbol{\Sigma}$ determine the direction and the corresponding eigenvalues determine the length of the principal axes. The quantity $r^2 = (\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is called the squared *Mahalanobis distance* from $\mathbf{x}$ to $\boldsymbol{\mu}$.

Gaussian Density Estimation

- The *maximum likelihood estimates* of a Gaussian are

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \quad \text{and} \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T.$$



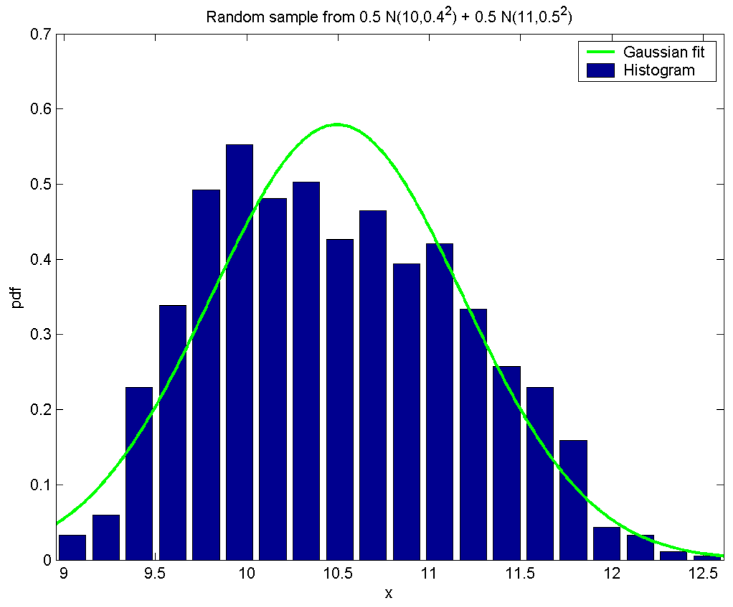


Figure 6: Gaussian density estimation examples.

Bernoulli Density Estimation

- Suppose that $P(x|\theta) = \text{Bernoulli}(\theta) = \theta^x(1-\theta)^{1-x}$ where $x = 0, 1$ and $0 \leq \theta \leq 1$.
- The maximum likelihood estimate of θ can be computed as

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Non-parametric Density Estimation

- Density estimation with parametric models assumes that the forms of the underlying density functions are known.
- However, common parametric forms do not always fit the densities actually encountered in practice.
- In addition, most of the classical parametric densities are unimodal, whereas many practical problems involve multimodal densities.
- Non-parametric methods can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known.

Histogram Method

- A very simple method is to partition the space into a number of equally-sized cells (*bins*) and compute a *histogram*.

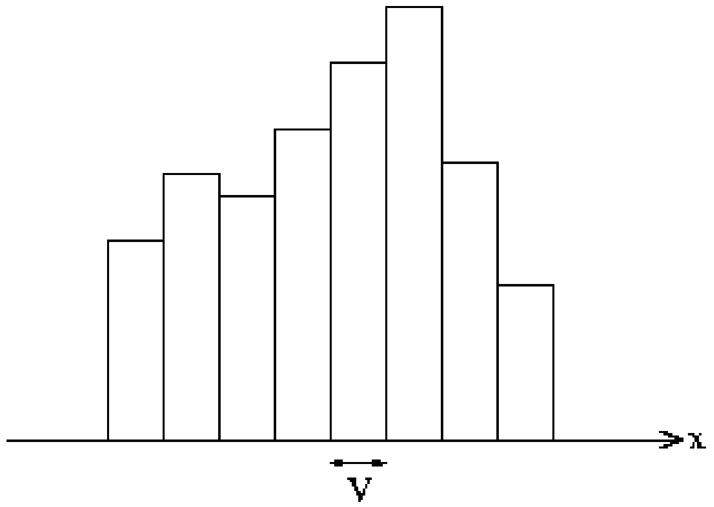


Figure 7: Histogram in one dimension.

- The estimate of the density at a point $\mathbf{x}$ becomes

$$p(\mathbf{x}) = \frac{k}{nV}$$

where n is the total number of samples, k is the number of samples in the cell that includes $\mathbf{x}$, and V is the volume of that cell.

Classification Error

- To apply these results to multiple classes, separate the training samples to c subsets $\mathcal{D}_1, \ldots, \mathcal{D}_c$, with the samples in $\mathcal{D}_i$ belonging to class w_i, and then estimate each density $p(\mathbf{x}|w_i, \mathcal{D}_i)$ separately.
- Different sources of error:
  - Bayes error: due to overlapping class-conditional densities (related to the features used).
  - Model error: due to incorrect model.
  - Estimation error: due to estimation from a finite sample (can be reduced by increasing the amount of training data).

Feature Reduction and Selection

- In practical multicategory applications, it is not unusual to encounter problems involving tens or hundreds of features.
- Intuitively, it may seem that each feature is useful for at least some of the discriminations.
- There are two issues that we must be careful about:
  - ▸ How is the classification accuracy affected by the dimensionality (relative to the amount of training data)?
  - ▸ How is the computational complexity of the classifier affected by the dimensionality?

Problems of Dimensionality

- In general, if the performance obtained with a given set of features is inadequate, it is natural to consider adding new features.
- Unfortunately, it has frequently been observed in practice that, beyond a certain point, adding new features leads to worse rather than better performance.
- This is called the *curse of dimensionality*.
- Potential reasons include wrong assumptions in model selection or estimation errors due to the finite number of training samples for high-dimensional observations (overfitting).

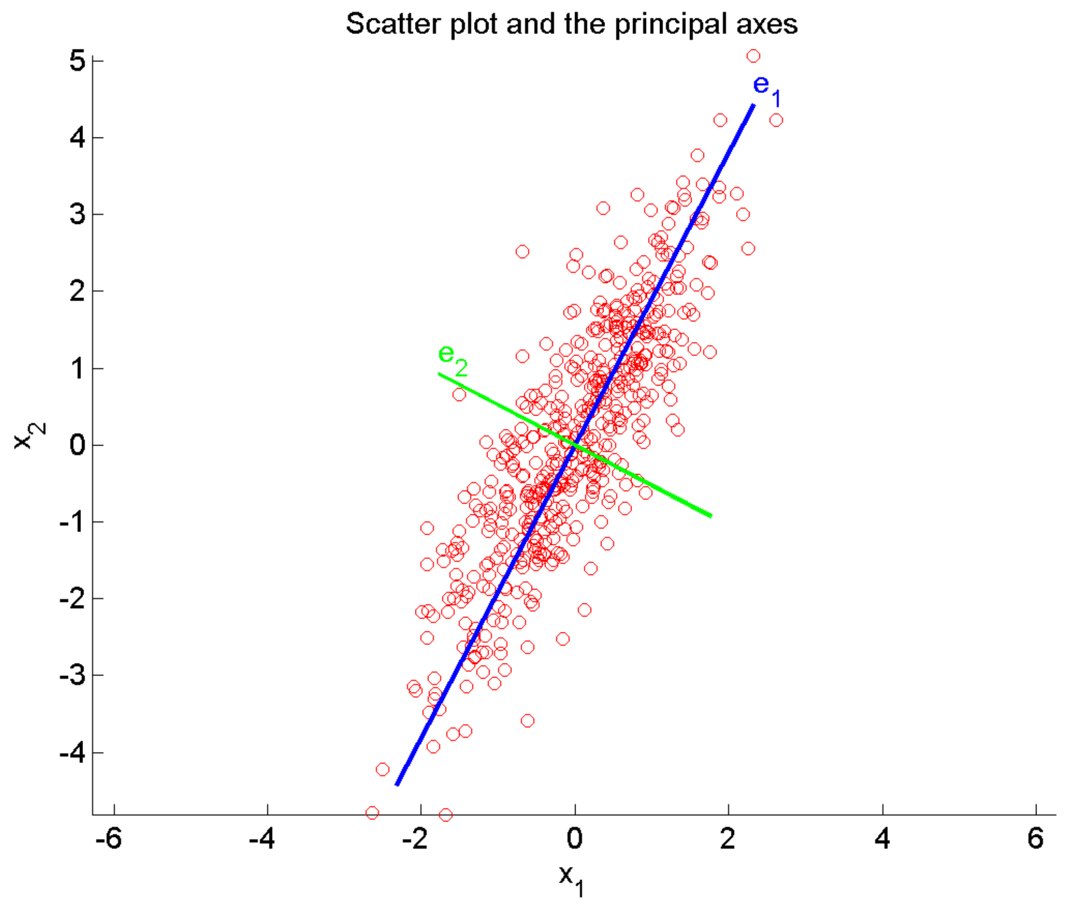
Problems of Dimensionality

- All of the commonly used classifiers can suffer from the curse of dimensionality.
- While an exact relationship between the probability of error, the number of training samples, the number of features, and the number of parameters is very difficult to establish, some guidelines have been suggested.
- It is generally accepted that using at least ten times as many training samples per class as the number of features ($n/d > 10$) is a good practice.
- The more complex the classifier, the larger should the ratio of sample size to dimensionality be.

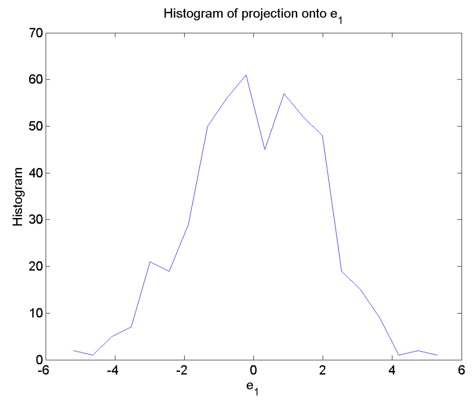
Problems of Dimensionality

- Dimensionality can be reduced by
  - redesigning the features
  - selecting an appropriate subset among the existing features
  - transforming to different feature spaces
    - *Principal Components Analysis (PCA)* seeks a projection that best represents the data in a least-squares sense.
    - *Linear Discriminant Analysis (LDA)* seeks a projection that best separates the data in a least-squares sense.

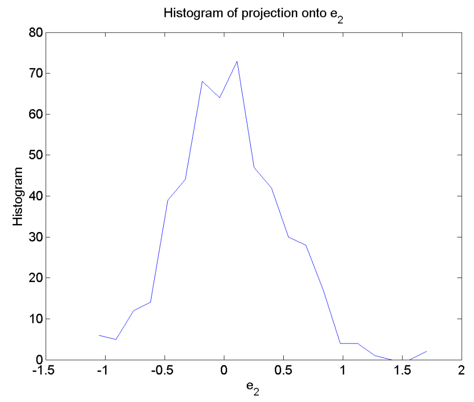
Examples



(a) Scatter plot.



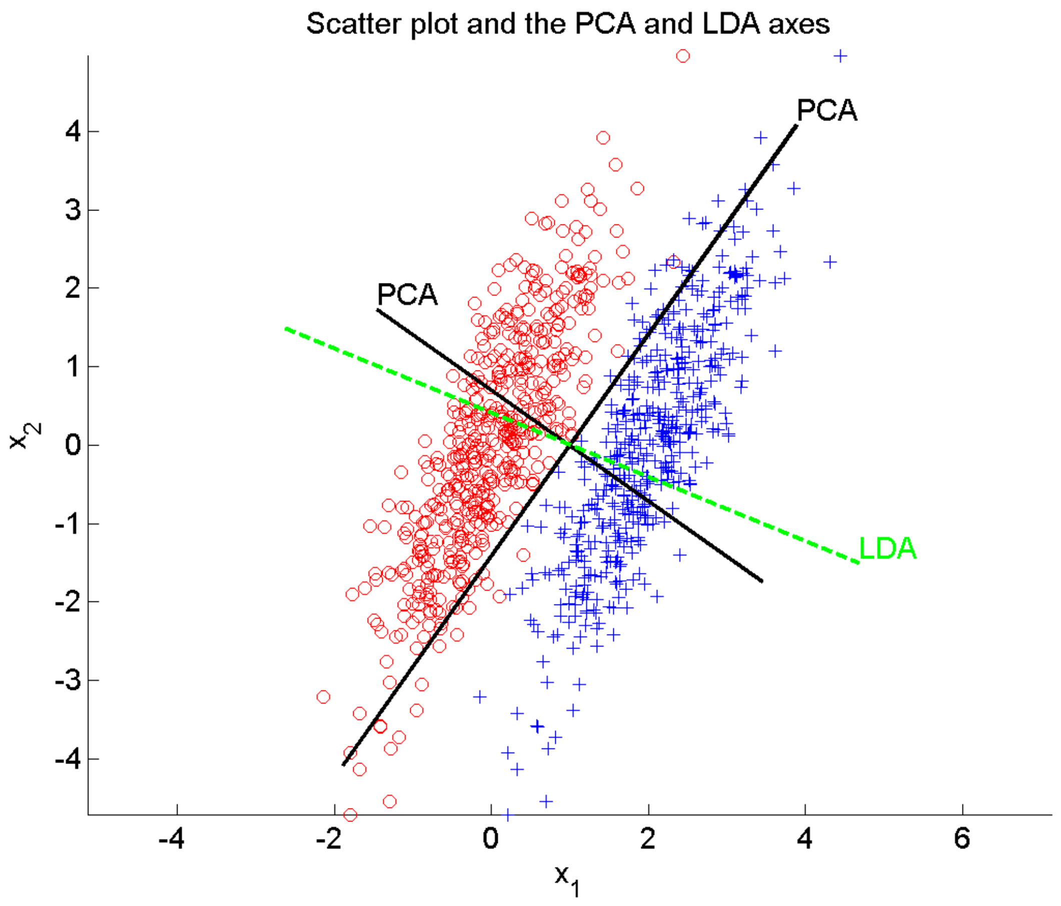
(b) Projection onto e_1.



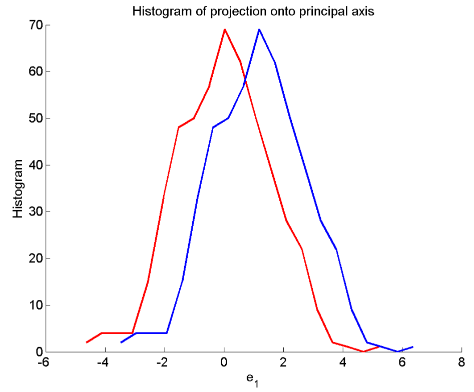
(c) Projection onto e_2.

Figure 8: Scatter plot (red dots) and the principal axes for a bivariate sample. The blue line shows the axis e_1 with the greatest variance and the green line shows the axis e_2 with the smallest variance. Features are now uncorrelated.

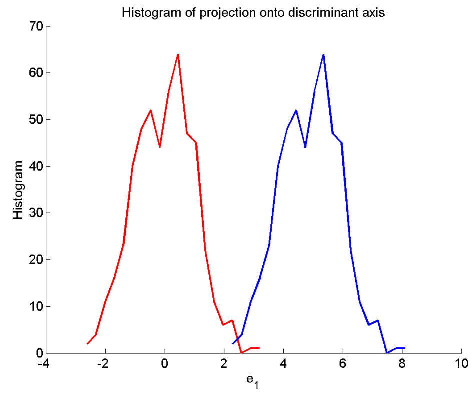
Examples



(a) Scatter plot.



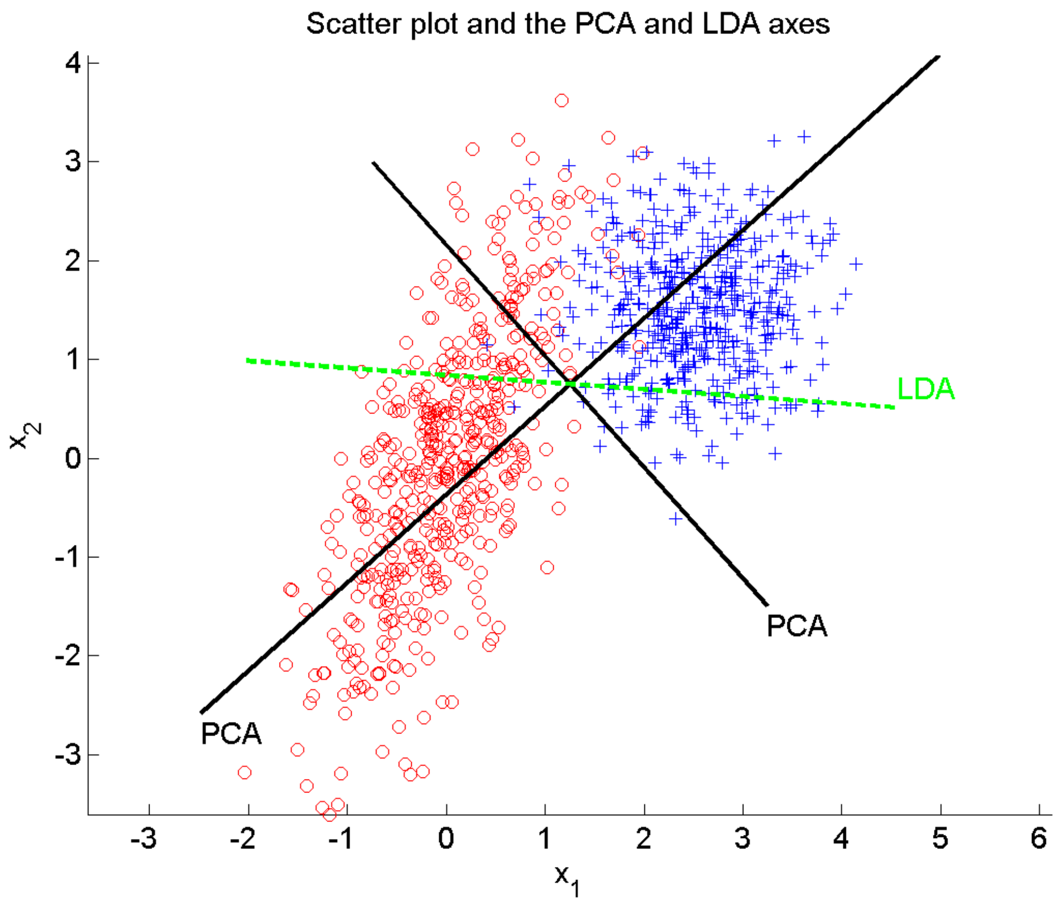
(b) Projection onto the first PCA axis.



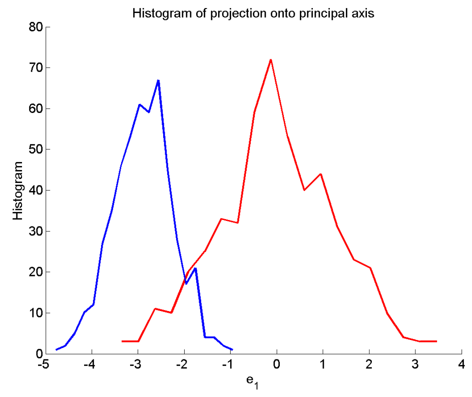
(c) Projection onto the first LDA axis.

Figure 9: Scatter plot and the PCA and LDA axes for a bivariate sample with two classes. Histogram of the projection onto the first LDA axis shows better separation than the projection onto the first PCA axis.

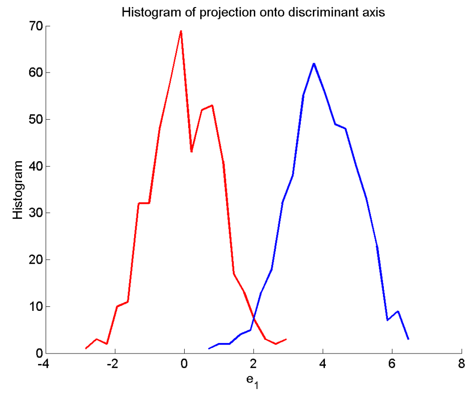
Examples



(a) Scatter plot.



(b) Projection onto the first PCA axis.



(c) Projection onto the first LDA axis.

Figure 10: Scatter plot and the PCA and LDA axes for a bivariate sample with two classes. Histogram of the projection onto the first LDA axis shows better separation than the projection onto the first PCA axis.

Non-Bayesian Classifiers

- Distance-based classifiers:
  - Minimum distance classifier
  - Nearest neighbor classifier
- Decision boundary-based classifiers:
  - Linear discriminant functions
  - Support vector machines
  - Neural networks
  - Decision trees

The k-Nearest Neighbor Classifier

- Given the training data $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ as a set of n labeled examples, the *nearest neighbor classifier* assigns a test point $\mathbf{x}$ the label associated with its closest neighbor in $\mathcal{D}$.
- The *k-nearest neighbor classifier* classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples.

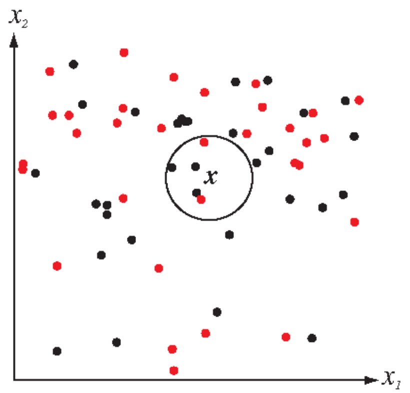


Figure 11: Classifier for $k = 5$.

- Closeness is defined using a distance function.

Distance Functions

- A general class of metrics for d-dimensional patterns is the *Minkowski metric*

$$L_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|^p \right)^{1/p}$$

  also referred to as the L_p *norm*.

- The *Euclidean distance* is the L_2 norm

$$L_2(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|^2 \right)^{1/2}$$

- The *Manhattan* or *city block distance* is the L_1 norm

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|$$

Distance Functions

- The L_∞ norm is the maximum of the distances along individual coordinate axes

$$L_\infty(\mathbf{x},\mathbf{y}) = \max_{i=1}^{d} |\mathbf{x}_i - \mathbf{y}_i|$$

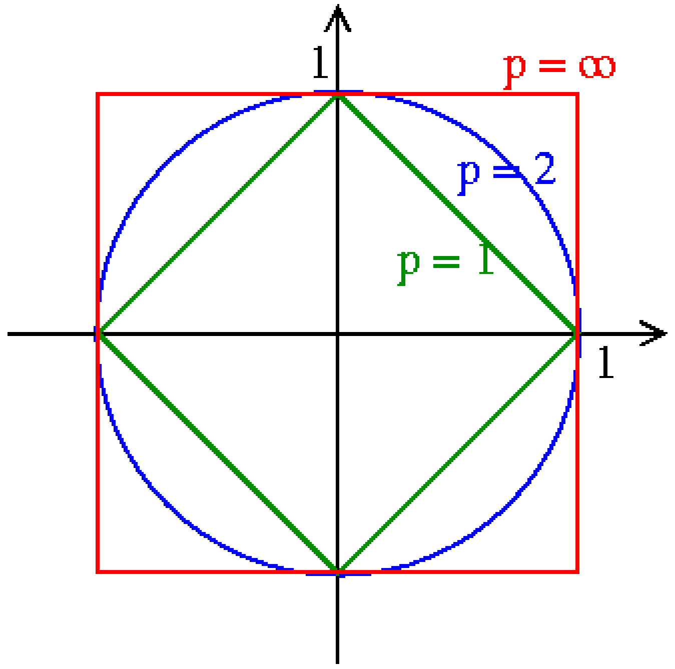


Figure 12: Each colored shape consists of points at a distance 1.0 from the origin, measured using different values of p in the Minkowski L_p metric.

Linear Discriminant Functions

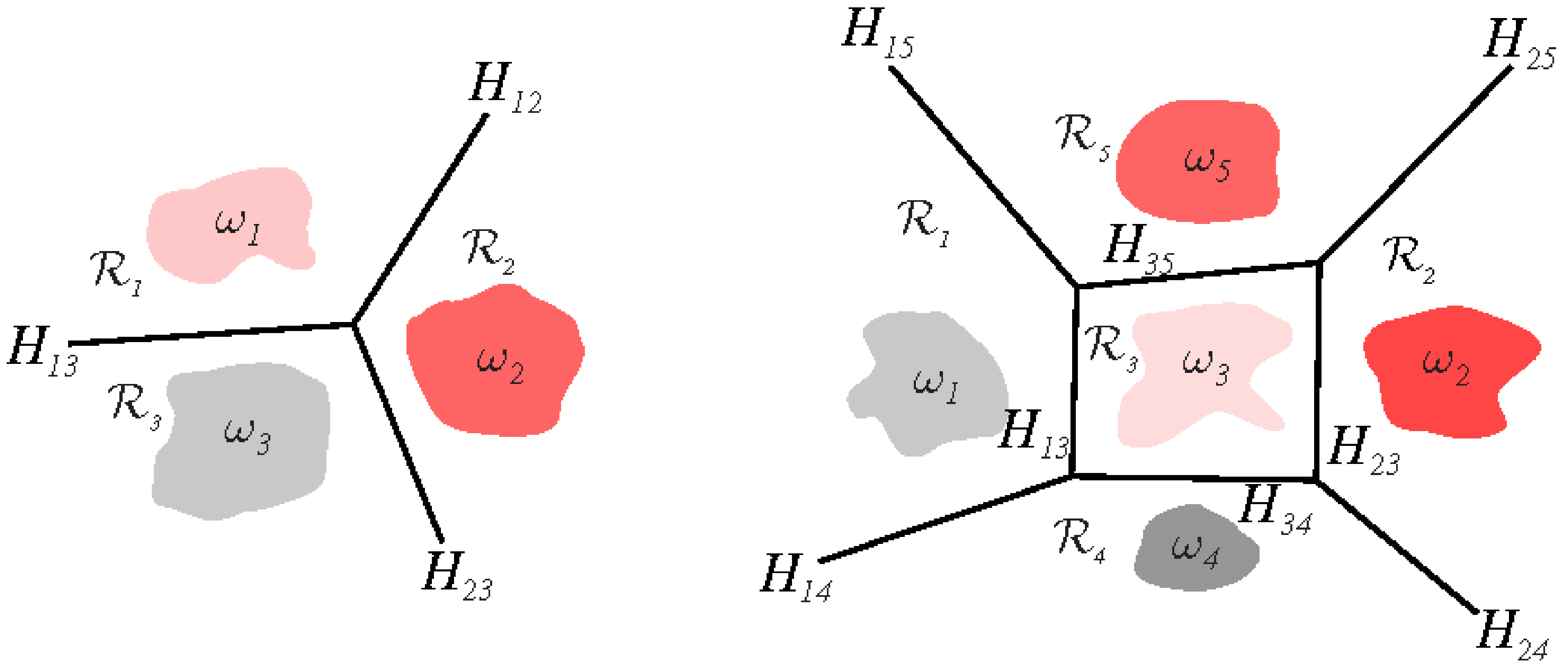


Figure 13: Linear decision boundaries produced by using one linear discriminant for each class.

Support Vector Machines

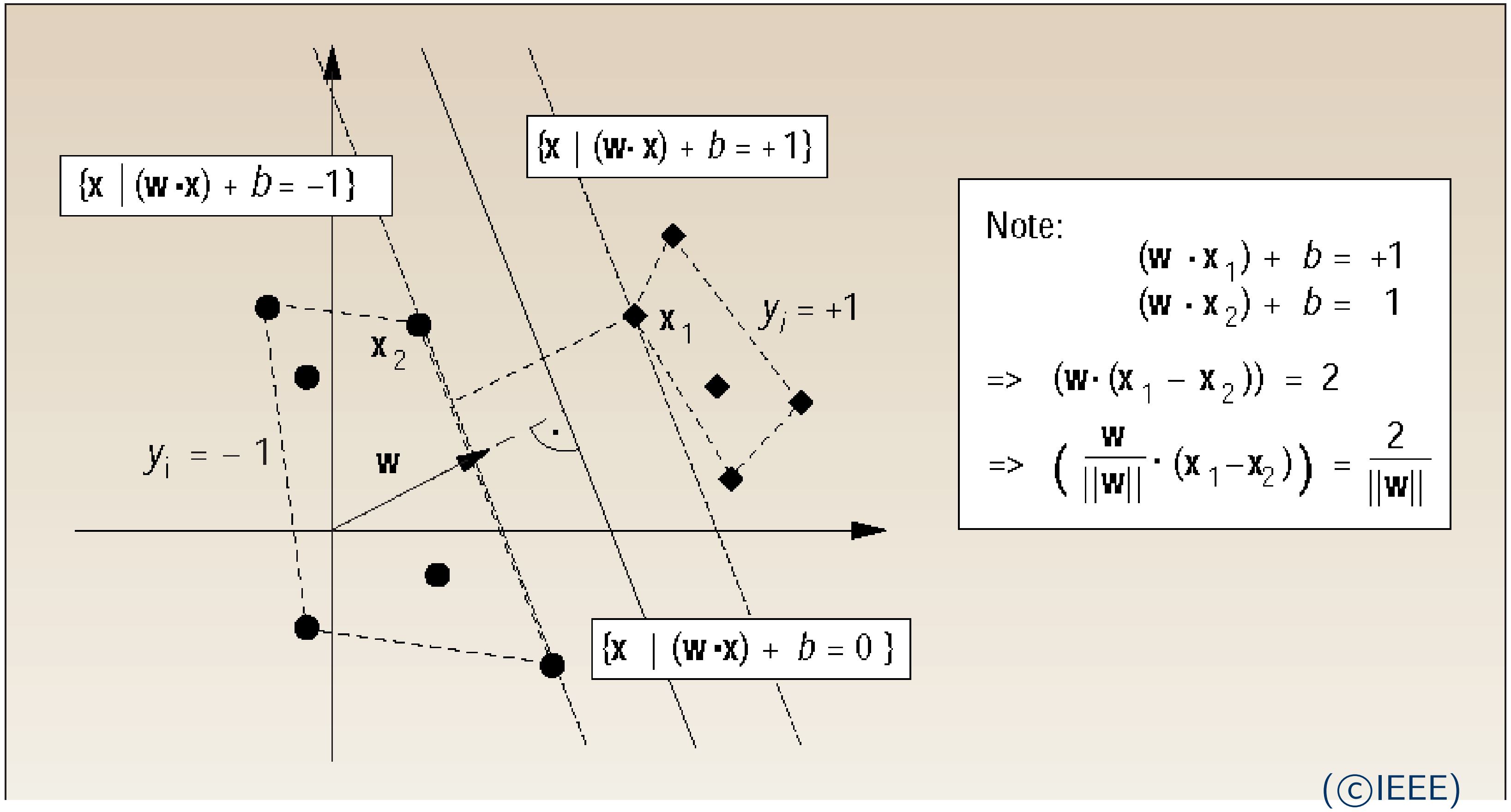
- Given a set of training patterns and class labels as $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n) \in \mathbb{R}^d \times \{\pm 1\}$, the goal is to find a classifier function $f : \mathbb{R}^d \rightarrow \{\pm 1\}$ such that $f(\mathbf{x}) = y$ will correctly classify new patterns.
- *Support vector machines* are based on the class of hyperplanes

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0, \quad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

corresponding to decision functions

$$f(\mathbf{x}) = \operatorname{sign}((\mathbf{w} \cdot \mathbf{x}) + b).$$

Support Vector Machines



$\{x \mid (w \cdot x) + b = -1\}$
$\{x \mid (w \cdot x) + b = +1\}$
$\{x \mid (w \cdot x) + b = 0\}$
x_2
x_1
$y_i = +1$
$y_i = -1$
w
Note:
$(w \cdot x_1) + b = +1$
$(w \cdot x_2) + b = 1$
$\Rightarrow (w \cdot (x_1 - x_2)) = 2$
$\Rightarrow \left(\frac{w}{\|w\|} \cdot (x_1 - x_2)\right) = \frac{2}{\|w\|}$
(©IEEE)

Figure 14: A binary classification problem of separating balls from diamonds. Support vector machines find hyperplane decision boundaries that yield the maximum margin of separation between the classes. The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted), and intersects it half way between the two classes.

Support Vector Machines

- To construct the optimal hyperplane, we can define the following optimization problem:

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to } y_i((\mathbf{w} \cdot \mathbf{x}_{\boldsymbol{i}}) + b) \geq 1, \quad i = 1, \ldots, n.$$

- This constrained optimization problem is solved using Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_{\boldsymbol{i}}) + b) - 1)$$

  where L has to be minimized w.r.t. the prime variables $\mathbf{w}$ and b, and maximized w.r.t. the dual variables α_i.

Support Vector Machines

- The solution can be obtained using quadratic programming techniques where the solution vector

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \, y_i \, \mathbf{x}_{\boldsymbol{i}}$$

is the summation of a subset of the training patterns, called the *support vectors*, whose α_i are non-zero.

- The support vectors lie on the margin and carry all relevant information about the classification problem (the remaining patterns are irrelevant).

Neural Networks

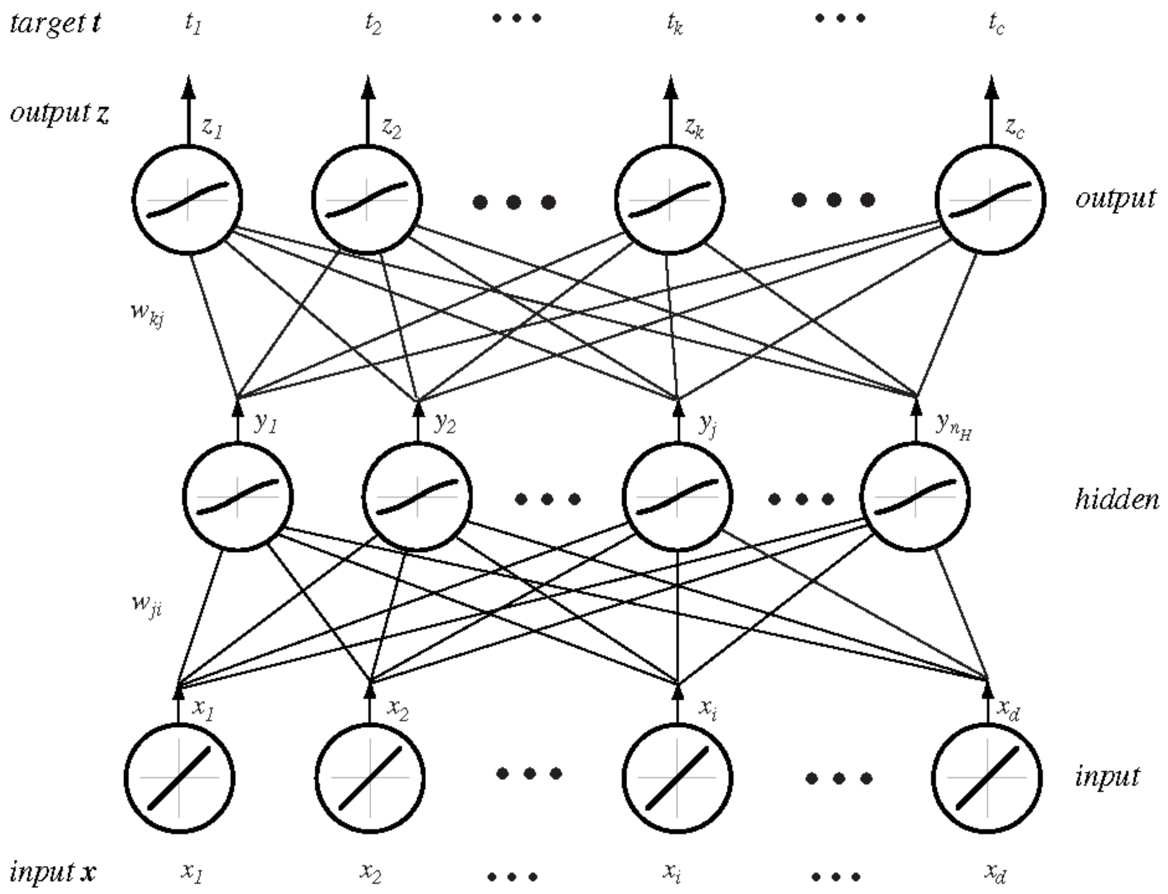


Figure 15: A neural network consists of an *input layer*, an *output layer* and usually one or more *hidden layers* that are interconnected by modifiable weights represented by links between layers. They learn the values of these weights as a mapping from the input to the output.

Decision Trees

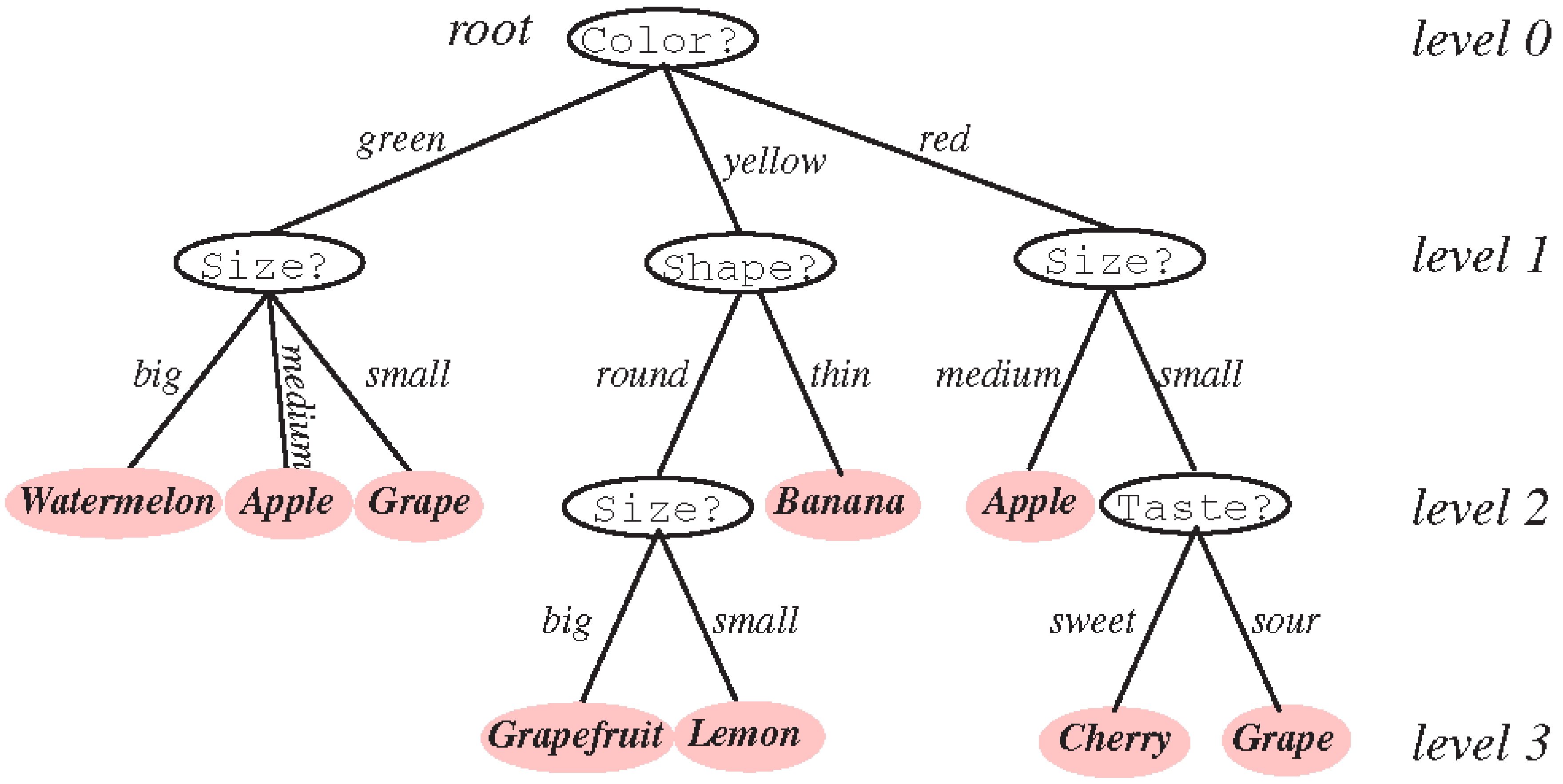


Figure 16: Decision trees classify a pattern through a sequence of questions, in which the next question asked depends on the answer to the current question.

Unsupervised Learning and Clustering

- *Clustering* is an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:
  - Collecting and labeling a large set of sample patterns can be costly or may not be feasible.
  - One can train with large amount of unlabeled data, and then use supervision to label the groupings found.
  - Unsupervised methods can be used for feature extraction.
  - Exploratory data analysis can provide insight into the nature or structure of the data.

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.
- Patterns within a cluster are more similar to each other than are patterns in different clusters.
- Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

Clustering

- Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes.

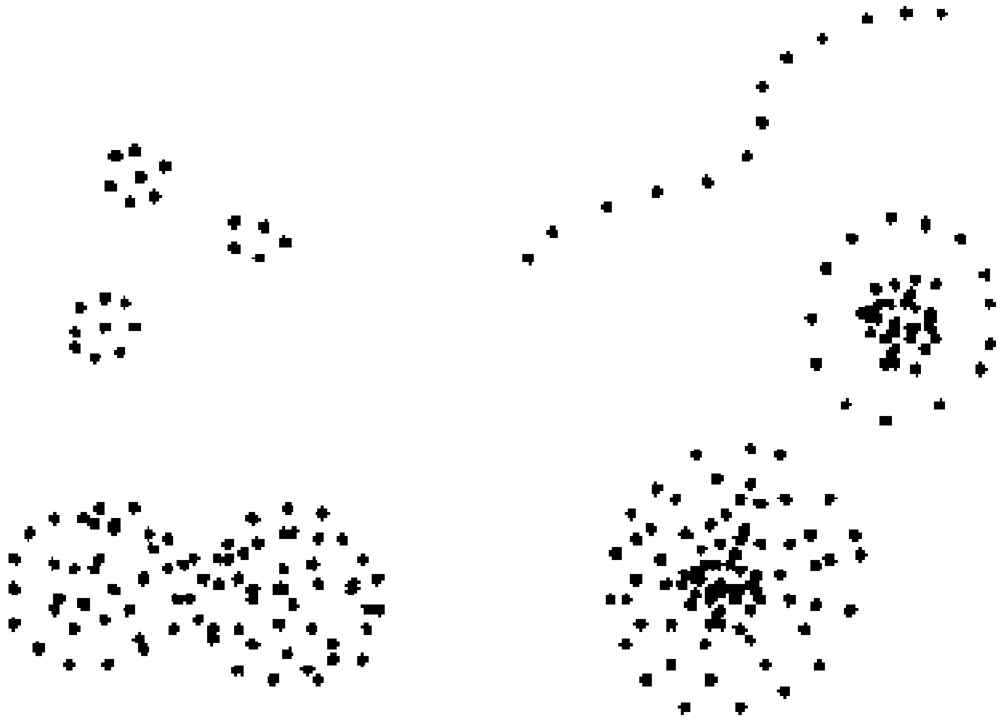


Figure 17: The number of clusters in the data often depend on the resolution (fine vs. coarse) with which we view the data. How many clusters do you see in this figure? 5, 8, 10, more?

Clustering

- Most of these algorithms are based on the following two popular techniques:
  - Iterative squared-error partitioning,
  - Agglomerative hierarchical clustering.
- One of the main challenges is to select an appropriate measure of similarity to define clusters that is often both data (cluster shape) and context dependent.

Squared-error Partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
- Let n_i be the number of samples in $\mathcal{D}_i$ and let $\mathbf{m}_i$ be the mean of those samples

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}.$$

- Then, the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2.$$

- For a given cluster $\mathcal{D}_i$, the mean vector $\mathbf{m}_i$ (centroid) is the best representative of the samples in $\mathcal{D}_i$.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
  1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
  2. Generate a new partition by assigning each pattern to its closest cluster center.
  3. Compute new cluster centers as the centroids of the clusters.
  4. Repeat steps 2 and 3 until an optimum value of the criterion function is found (e.g., when a local minimum is found or a predefined number of iterations are completed).
  5. Adjust the number of clusters by merging and splitting existing clusters or by removing small or outlier clusters.
- This algorithm, without step 5, is also known as the *k-means* algorithm.

Squared-error Partitioning

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape and well-separated in the feature space.
- However, choosing k and choosing the initial partition are the main drawbacks of this algorithm.
- The value of k is often chosen empirically or by prior knowledge about the data.
- The initial partition is often chosen by generating k random points uniformly distributed within the range of the data, or by randomly selecting k points from the data.

Hierarchical Clustering

- The k-means algorithm produces a *flat* data description where the clusters are disjoint and are at the same level.
- In some applications, groups of patterns share some characteristics when looked at a particular level.
- Hierarchical clustering tries to capture these multi-level groupings using *hierarchical* representations rather than flat partitions.

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),
  - the second level contains $n - 1$ clusters,
  - the third level contains $n - 2$ clusters,
  - and so on until the last (n'th) level at which all samples form a single cluster.
- Given any two samples, at some level they will be grouped together in the same cluster and remain together at all higher levels.

Hierarchical Clustering

- A natural representation of hierarchical clustering is a tree, also called a *dendrogram*, which shows how the samples are grouped.
- If there is an unusually large gap between the similarity values for two particular levels, one can argue that the level with fewer number of clusters represents a more natural grouping.

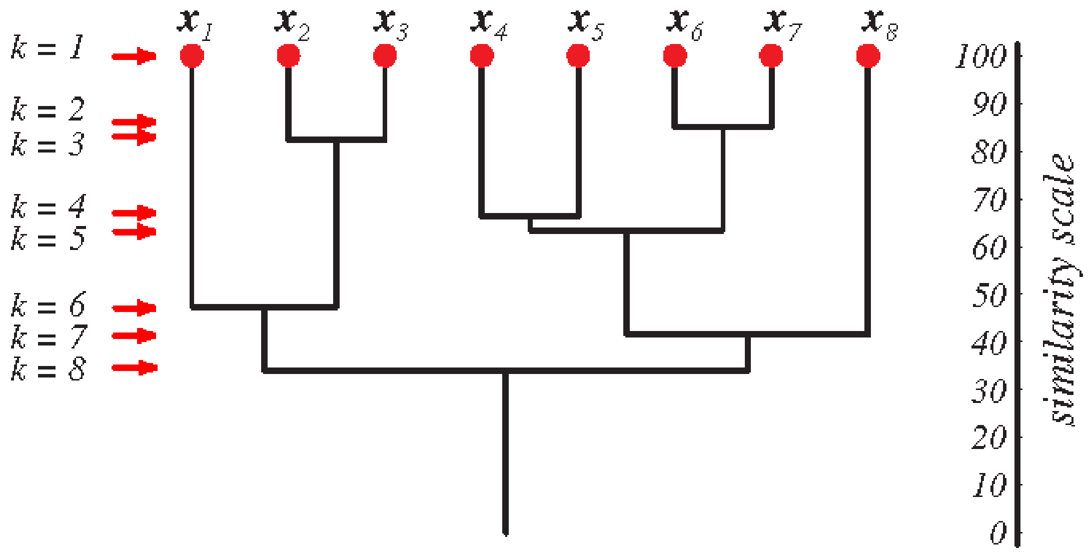


Figure 18: A dendrogram can represent the results of hierarchical clustering algorithms.

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the closest clusters according to a distance measure.
  3. Merge these two clusters.
  4. Return the resulting clusters.

Hierarchical Clustering

- Popular distance measures (for two clusters $\mathcal{D}_i$ and $\mathcal{D}_j$):

$$d_{\mathsf{min}}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\mathsf{max}}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\mathsf{avg}}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{\#\mathcal{D}_i \, \#\mathcal{D}_j} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\mathsf{mean}}(\mathcal{D}_i, \mathcal{D}_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

Algorithm-Independent Learning Issues

- We have seen many learning algorithms and techniques for pattern recognition.
- Some of these algorithms may be preferred because of their lower computational complexity.
- Others may be preferred because they take into account some prior knowledge of the form of the data.
- Given practical constraints such as finite training data, no pattern classification method is inherently superior to any other.

Estimating and Comparing Classifiers

- Classification error can be estimated using misclassification and false alarm rates.
- To compare learning algorithms, we should use independent training and test data generated using
  - static division,
  - rotated division (e.g., cross-validation),
  - bootstrap methods.
- Using the error on points not in the training set (also called the *off-training set error*) is important for evaluating the generalization ability of an algorithm.

Combining Classifiers

- Just like different features capturing different properties of a pattern, different classifiers also capture different structures and relationships of these patterns in the feature space.
- An empirical comparison of different classifiers can help us choose one of them as the best classifier for the problem at hand.
- However, although most of the classifiers may have similar error rates, sets of patterns misclassified by different classifiers do not necessarily overlap.
- Not relying on a single decision but rather combining the advantages of different classifiers is intuitively promising to improve the overall accuracy of classification.
- Such combinations are variously called *combined classifiers*, *ensemble classifiers*, *mixture-of-expert models*, or *pooled classifiers*.

Combining Classifiers

- Some of the reasons for combining multiple classifiers to solve a given classification problem can be stated as follows:
  - Access to different classifiers, each developed in a different context and for an entirely different representation/description of the same problem.
  - Availability of multiple training sets, each collected at a different time or in a different environment, even may use different features.
  - Local performances of different classifiers where each classifier may have its own region in the feature space where it performs the best.
  - Different performances due to different initializations and randomness inherent in the training procedure.

Combining Classifiers

- In summary, we may have different feature sets, training sets, classification methods, and training sessions, all resulting in a set of classifiers whose outputs may be combined.
- Combination architectures can be grouped as:
  - *Parallel:* all classifiers are invoked independently and then their results are combined by a combiner.
  - *Serial (cascading):* individual classifiers are invoked in a linear sequence where the number of possible classes for a given pattern is gradually reduced.
  - *Hierarchical (tree):* individual classifiers are combined into a structure, which is similar to that of a decision tree, where the nodes are associated with the classifiers.

Combining Classifiers

- Examples of classifier combination schemes are:
  - *Majority voting* where each classifier makes a binary decision (vote) about each class and the final decision is made in favor of the class with the largest number of votes.
  - *Bayesian combination:* sum, product, maximum, minimum and median of the posterior probabilities from individual classifiers.
  - *Bagging* where multiple classifiers are built by bootstrapping the original training set.
  - *Boosting* where a sequence of classifiers is built by training each classifier using data sampled from a distribution derived from the empirical misclassification rate of the previous classifier.

Structural and Syntactic Pattern Recognition

- Statistical pattern recognition attempts to classify patterns based on a set of extracted features and an underlying statistical model for the generation of these patterns.
- Ideally, this is achieved with a rather straightforward procedure:
  - determine the feature vector,
  - train the system,
  - classify the patterns.
- Unfortunately, there are also many problems where patterns contain structural and relational information that are difficult or impossible to quantify in feature vector form.

Structural and Syntactic Pattern Recognition

- *Structural pattern recognition* assumes that pattern structure is quantifiable and extractable so that structural similarity of patterns can be assessed.
- Typically, these approaches formulate hierarchical descriptions of complex patterns built up from simpler primitive elements.
- This structure quantification and description are mainly done using:
  - Formal grammars,
  - Relational descriptions (principally graphs).
- Then, recognition and classification are done using:
  - Parsing (for formal grammars),
  - Relational graph matching (for relational descriptions).