Supervised Classification

- Introduction
- Bayesian Classification
- Minimum Risk Theory
- Discriminant Functions
- Decision Trees
- Accuracy Evaluation

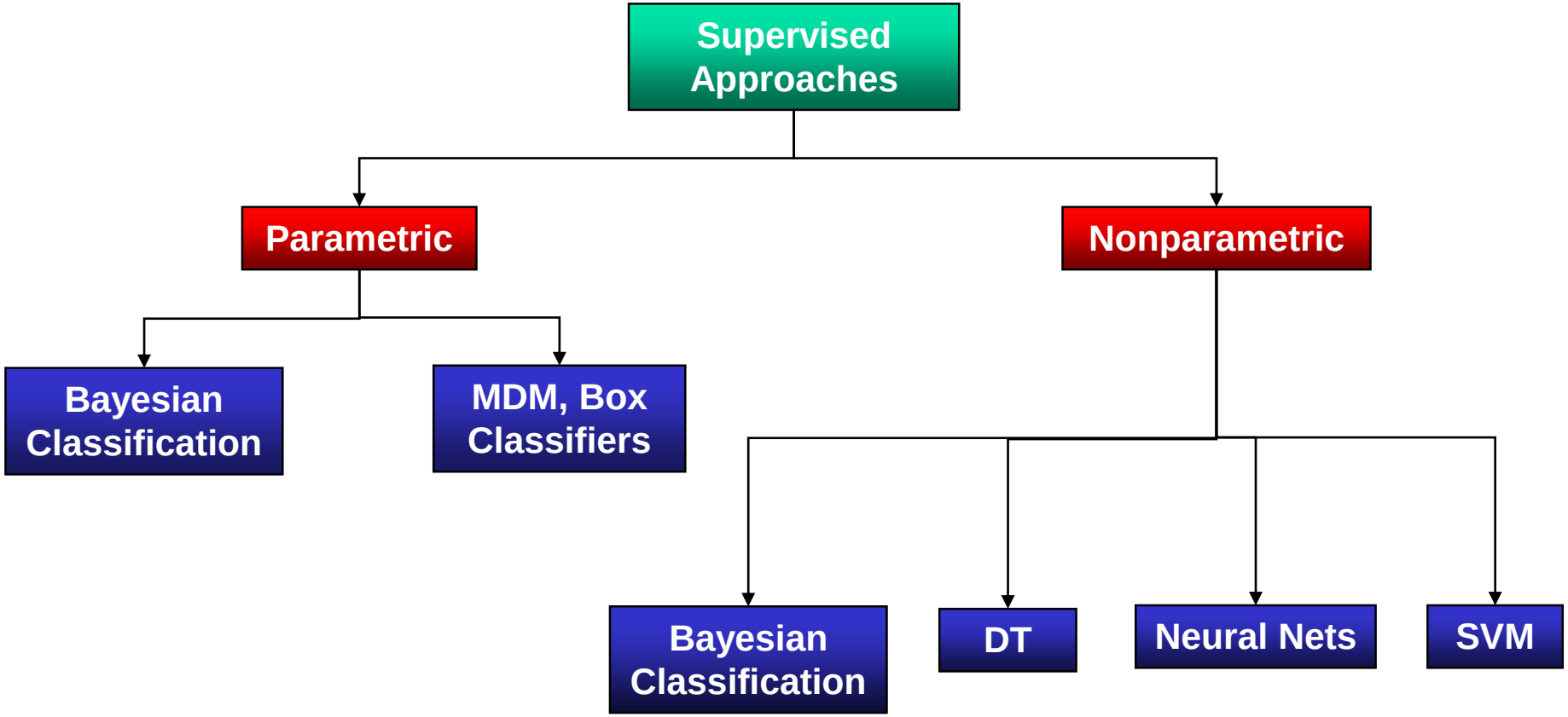
Introduction

- Depending on the availability or not of training samples, one may resort to supervised or unsupervised machine learning techniques.
- In this chapter, we will deal with the first category of techniques.
- The design of a supervised classifier depends on:
  - the available set of features;
  - the available set of training samples;
  - the typology of the adopted classification model;
  - the cost function to optimize.

Introduction

- Let $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ be a vector defined in a n-dimensional feature space.
- Let Ω be a set of C classes ω_i ($i=1, 2, \ldots, C$).
- Let $\mathbf{X}$ be a set of N training samples.
- The availability of training samples permits integrating prior knowledge of the problem in the classifier design through the definition of a statistical model for each class.
- Supervised classification aims at associating to each pattern $\mathbf{x}$ a class that optimizes a predefined decision criterion computed on the basis of the class models.

Introduction



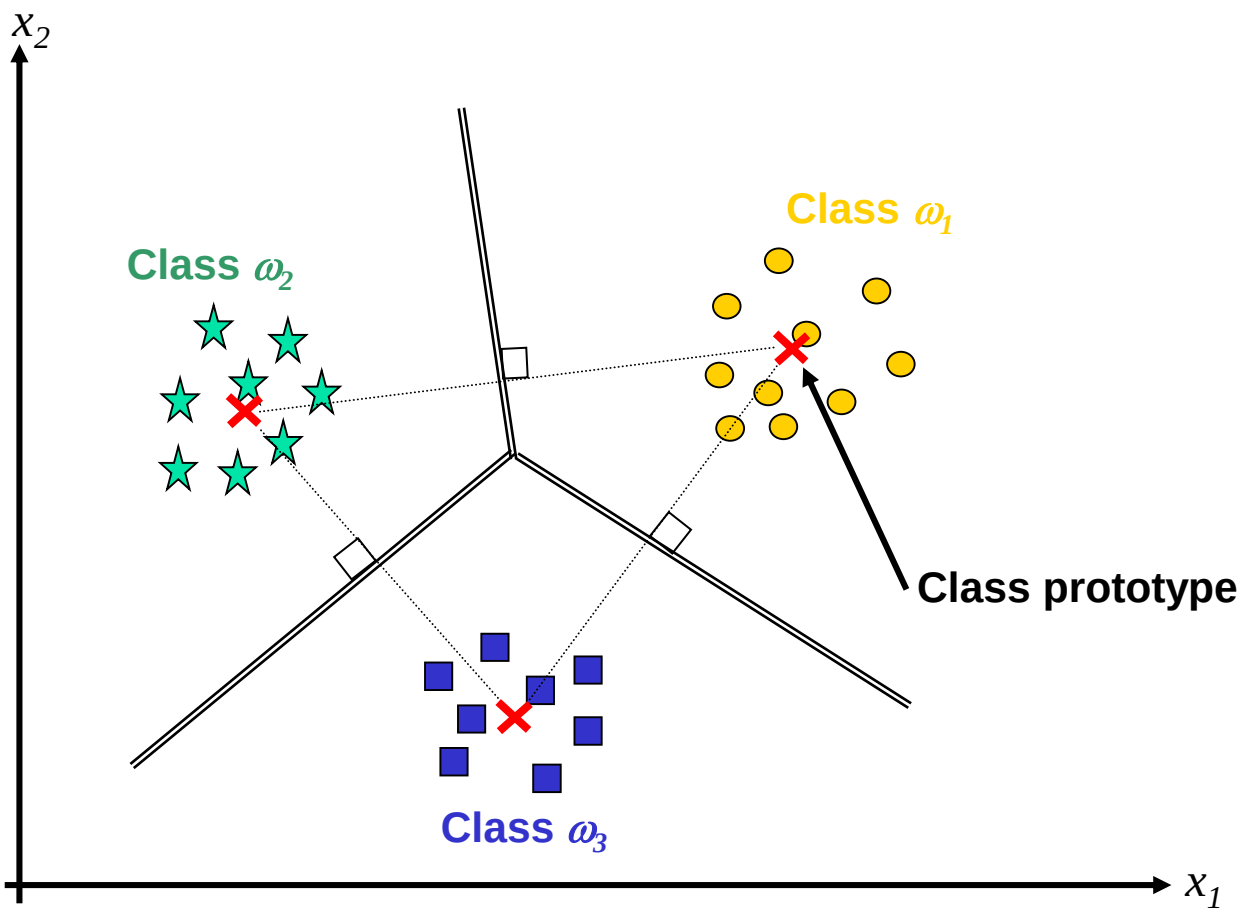
MDM Classifier

- **Underlying Hypotheses**:
  - Classes should be characterized by a small sample dispersion around the barycenter;

  or

  - Classes should have the same statistical behavior (i.e., the same Σ).
- The minimal-distance-to-means (MDM) classifier models each class just through its barycenter (which thus plays the role of class prototype).
- The classification of a given unlabeled pattern is done by:
  1) computing the distance between it and each of the C class prototypes;
  2) and assigning it to the nearest class.
- MDM classification partitions the feature space with linear frontiers.

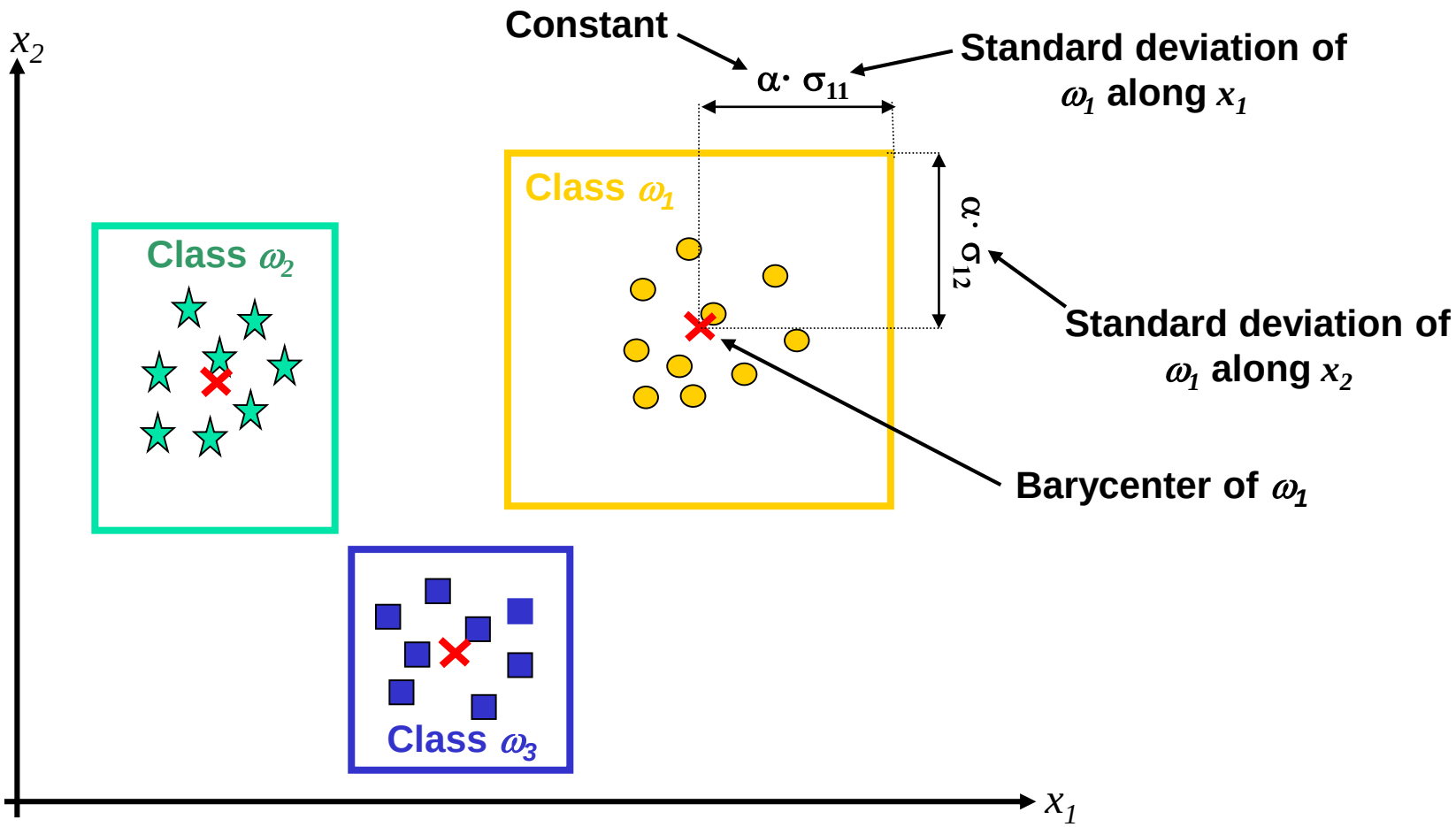
MDM Classifier: Example



Box Classifier

- Unlike the MDM classifier, the Box classifier exploits 2° order statistics (though in a primitive way).
- It models each class with a uniform pdf:
  - centered on the class barycenter;
  - with a size proportional to the standard deviation along each dimension of the feature space.
- In practice, the resulting boxes can be seen as decision regions.
- A given unknown pattern is classified according to the box it belongs to.

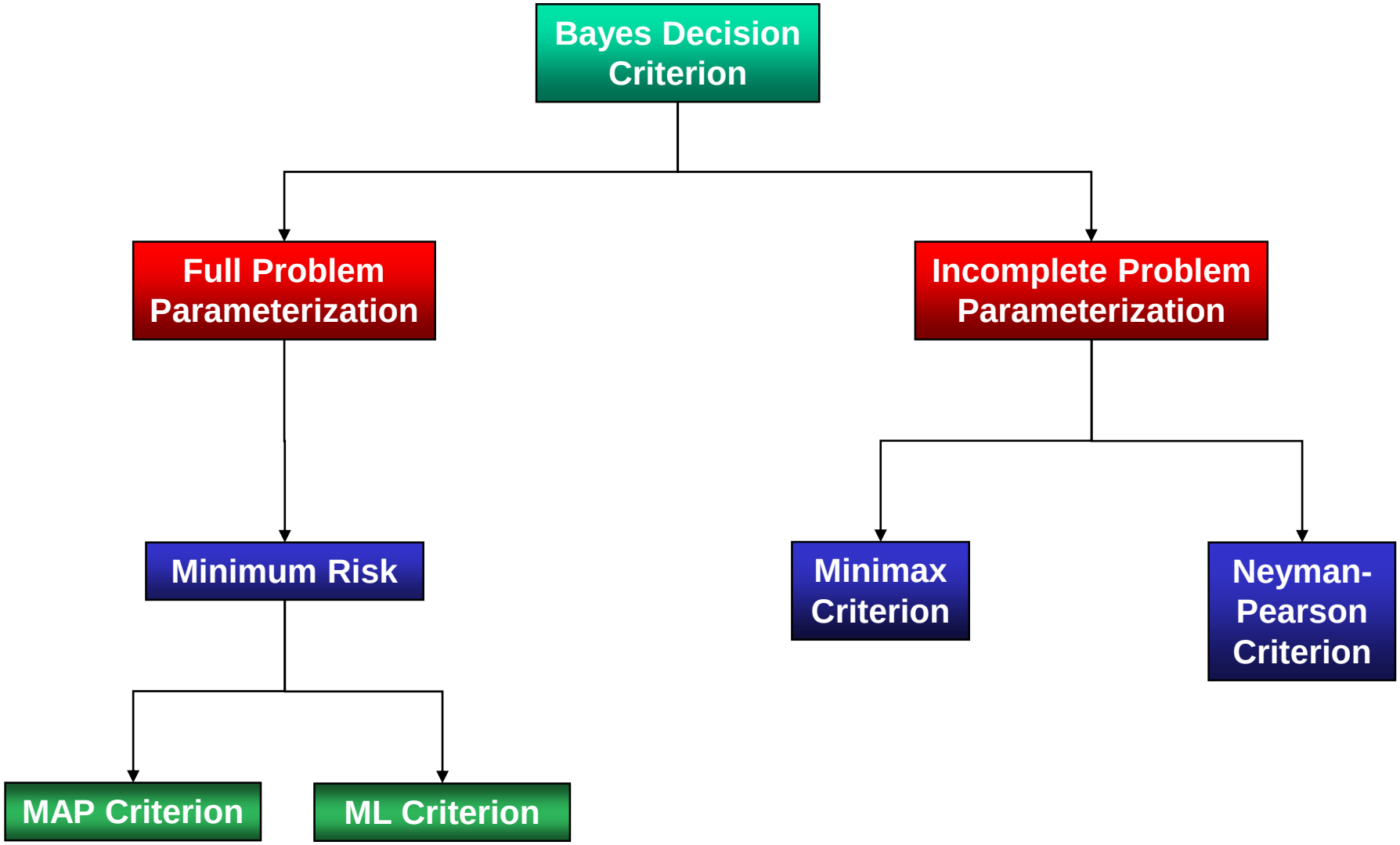
Box Classifier: Example



Bayesian Classification

- Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification.
- This approach is based on quantifying the tradeoffs between various classification decisions using probability and the costs that accompany such decisions.
- It makes the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known.
- Depending on the peculiarities and requirements of the considered classification problem, different decision criteria may be considered.

Bayesian Classification



MAP Criterion

- **Hypothesis**: a posteriori (posterior) probabilities of classes $P(\omega_i|\mathbf{x})$ $(i = 1, 2, ..., C)$ are assumed to be known.
- **Goal**: minimize the average probability of error.
- **Decision rule**: a pattern $\mathbf{x}$ is assigned to the class that maximizes the a posteriori probability $P(\omega_i|\mathbf{x})$:

$$\mathbf{x} \in \omega_j \quad \Longleftrightarrow \quad P(\omega_j|\mathbf{x}) \geq P(\omega_i|\mathbf{x}),\ \forall\ i = 1, 2, ..., C$$

- Since the posterior probabilities are often not directly known, it is preferable to rewrite the MAP decision rule by using the Bayes theorem $(posterior = \frac{likelihood \times prior}{evidence})$ as follows:

$$\mathbf{x} \in \omega_j \quad \Longleftrightarrow \quad P(\omega_j)p(\mathbf{x}|\omega_j) \geq P(\omega_i)p(\mathbf{x}|\omega_i), \quad \forall\ i = 1, 2, ..., C$$

MAP Criterion

- Let $\Re_i$ be the decision region generated by an arbitrary classifier for the class ω_i.

- The average probability of error is given by:

$$\mathrm{P_e} = \sum_{i=1}^{C} P(err|\omega_i)P(\omega_i) = \sum_{i=1}^{C} P(\mathrm{x} \notin \Re_i|\omega_i)P(\omega_i)$$

ML Criterion

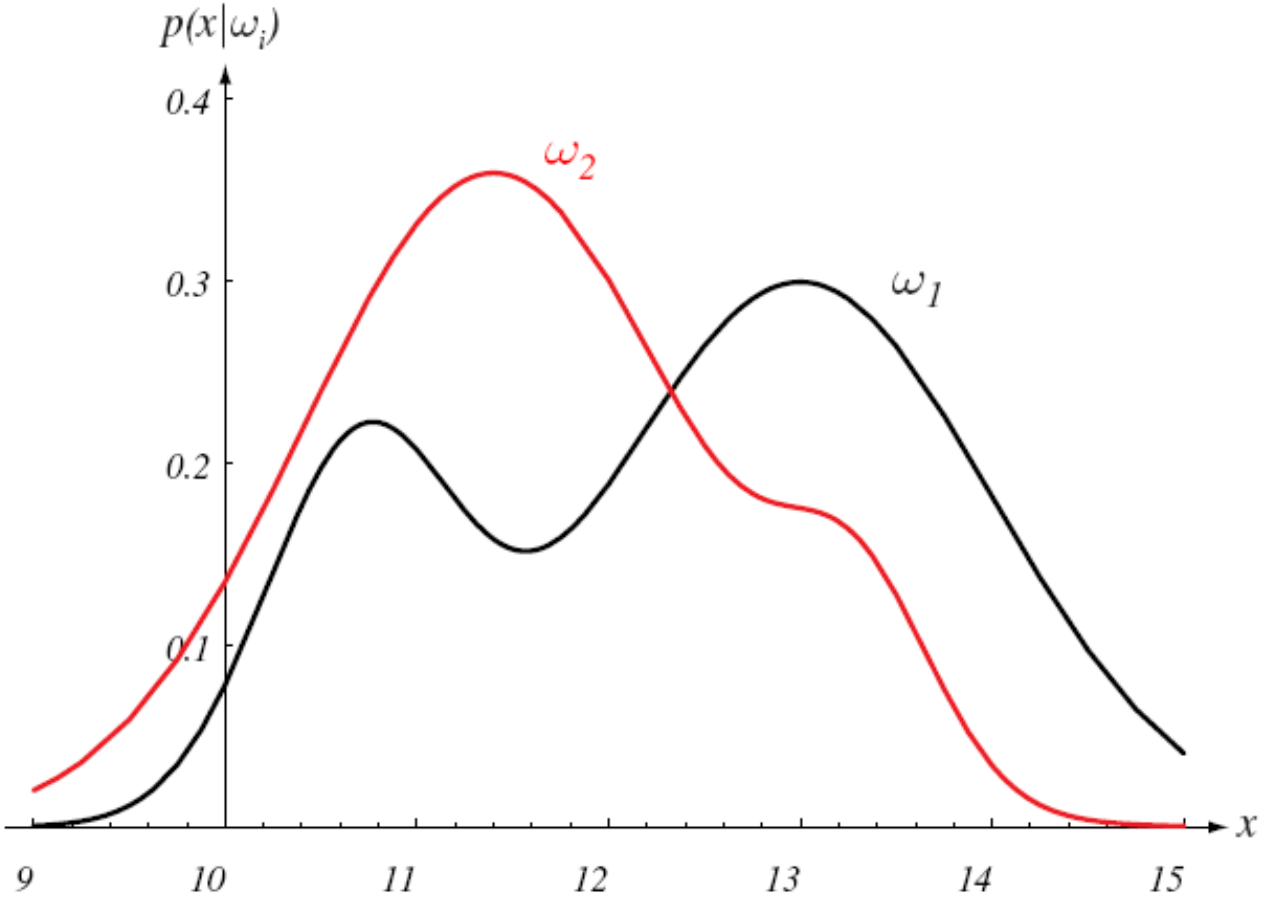
- **Hypothesis**: only the class-conditional pdfs $p(\mathbf{x}|\omega_i)$ $(i=1,2,...,C)$ are assumed to be known.
- **Goal**: minimize the average probability of error.
- **Decision rule**:

$$\mathbf{x} \in \omega_j \quad \Longleftrightarrow \quad p(\mathbf{x}|\omega_j) \geq p(\mathbf{x}|\omega_i),\ \forall\ i = 1, 2, ..., C$$

- Note that if the priors are equal, the ML criterion is equivalent to the optimal MAP criterion.

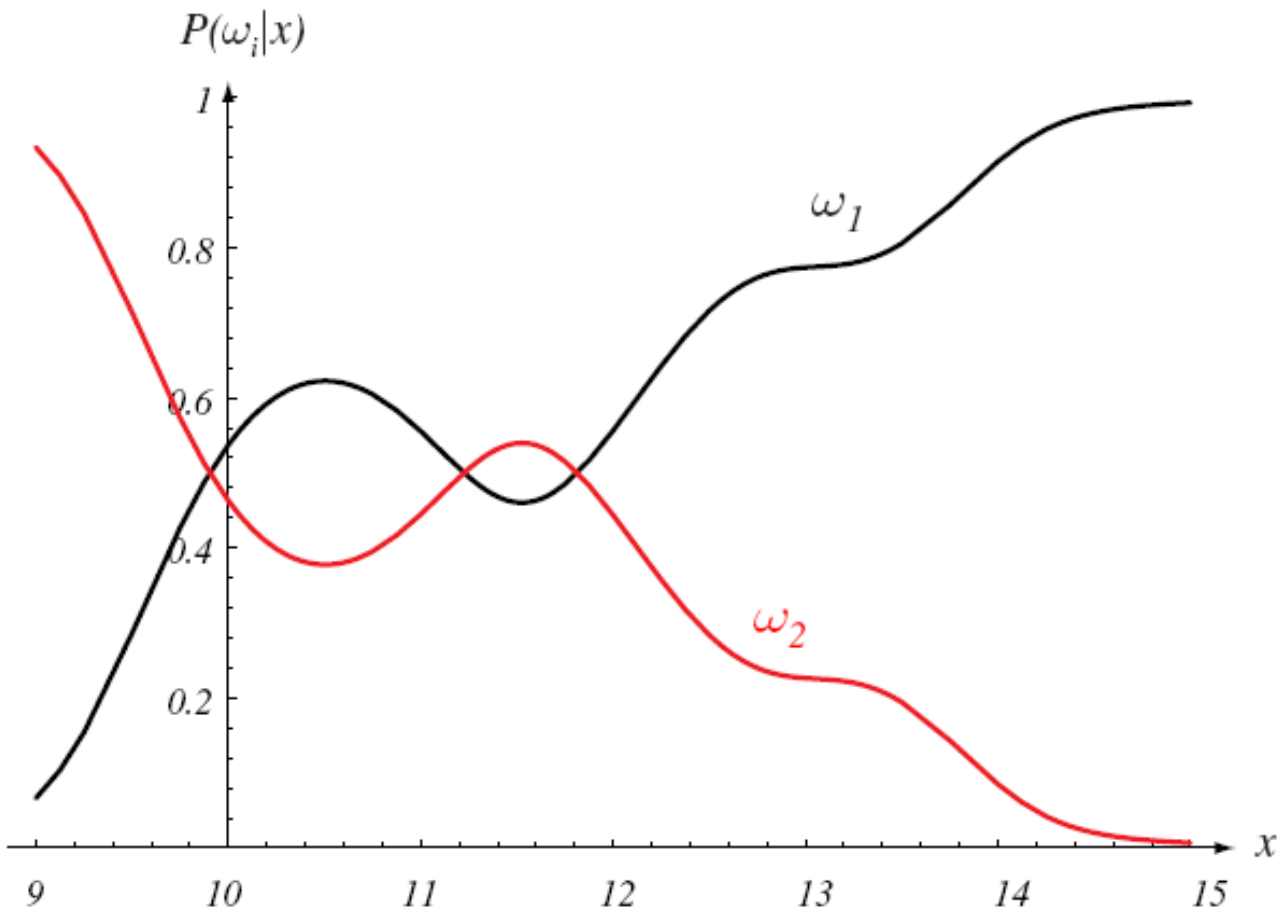
MAP-ML: Example

1-D class-conditional probability density functions for two classes.



MAP-ML: Example

Corresponding posterior probabilities for the particular priors $P(\omega_1) = 2/3$ and $P(\omega_2) = 1/3$.



Minimum Risk Theory

- Minimum risk theory is a general decision theory which includes as particular cases the MAP and the ML decision rules.
- These two rules minimize the average probability of error, without taking care about the costs incurred by the different classification errors.
- Indeed, in some applications, any decision is followed by an action with predefined cost.
- Therefore, the correct or wrong classification of a given observation may induce different costs.
- Minimum risk theory allows integrating information about error costs in the decision process.
- Let A= $\{\alpha_1, \alpha_2, ..., \alpha_R\}$ be the set of R possible actions.

Minimum Risk Theory

- The cost of an action depends on the true class to which a considered observation belongs.
- It is thus possible to define a $R \times C$ cost matrix Λ:

$$\Lambda = \begin{bmatrix} \lambda(\alpha_1|\omega_1) & \lambda(\alpha_1|\omega_2) & . & . & . & \lambda(\alpha_1|\omega_C) \\ \lambda(\alpha_2|\omega_1) & \lambda(\alpha_2|\omega_2) & . & & & \lambda(\alpha_2|\omega_C) \\ . & & & . & & . \\ . & & & & . & . \\ \lambda(\alpha_R|\omega_1) & \lambda(\alpha_R|\omega_2) & . & . & . & \lambda(\alpha_R|\omega_C) \end{bmatrix}$$

$\lambda_{ij} = \lambda(\alpha_i|\omega_j)$ ($\lambda_{ij} \geq 0$) is the cost (loss) incurred for taking action α_i given the class is ω_j.

Minimum Risk Theory

Example:

- $\Omega = \{\omega_1$="fire", ω_2="no fire"$\}$
- $\Lambda = \{\alpha_1$="do call firemen", α_2="do not call firemen"$\}$

$$\Lambda = \begin{bmatrix} 0 & \alpha_1 \\ \alpha_2 & 0 \end{bmatrix}$$

α_1 is the cost to call firemen when no fire has occurred and α_2 the loss incurred to not call them when fire actually occurred (e.g., $\alpha_1=10^3$ euros and $\alpha_2=10^6$ euros).

Minimum Risk Theory

Conditional Risk:

- For each pattern $\mathbf{x}$, let us introduce the conditional risk (expected loss) of taking action α_i:

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{C} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x})$$

Bayes Risk:

- Given $\mathbf{x}$, the action α_j which minimizes the conditional risk is chosen:

$$\mathbf{x}\,\alpha \quad \alpha^* = \alpha_j \quad \Longleftrightarrow \quad R(\alpha_j|\mathbf{x}) \le R(\alpha_i|\mathbf{x}), \quad \forall i = 1,2,\ldots\ldots,R$$

- The resulting overall risk

$$R^* = \int_{\mathbf{x}} R(\alpha^*|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

is called the Bayes risk and is the best performance that can be achieved.

Minimum Risk Theory: Example

- Let us consider the particular case where $C=R=2$.
- Λ is thus a 2×2 square matrix:

$$\begin{cases} R(\alpha_1|\mathbf{x}) = P(\omega_1|\mathbf{x})\lambda_{11} + P(\omega_2|\mathbf{x})\lambda_{12} \\ R(\alpha_2|\mathbf{x}) = P(\omega_1|\mathbf{x})\lambda_{21} + P(\omega_2|\mathbf{x})\lambda_{22} \end{cases}$$

- Given $\mathbf{x}$, action α_1 is chosen iff:

$$R(\alpha_1|\mathbf{x}) \leq R(\alpha_2|\mathbf{x})$$

Minimum Risk Theory: Example

- The decision rule can be rewritten as:

$$L(\mathbf{x}) = \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} \geq \frac{P(\omega_2)}{P(\omega_1)} \frac{(\lambda_{12} - \lambda_{22})}{(\lambda_{21} - \lambda_{11})} \qquad (\text{for } \lambda_{21} - \lambda_{11} > 0)$$

- The hypothesis $\lambda_{21}-\lambda_{11}>0$ is generally verified since, when Λ is square, the actions with lowest loss are those corresponding to the matrix diagonal which are interpreted as correct actions.

Minimum Risk Theory: Example

- Minimum risk criterion: if $\lambda_{22}=\lambda_{11}=0$ (zero cost for correct actions), we get

$$\mathbf{x} \;\alpha\; \alpha_1 \qquad \Lambda(\mathbf{x}) = \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} \geq \frac{P(\omega_2)}{P(\omega_1)}\frac{\lambda_{12}}{\lambda_{21}}$$

- MAP criterion:

$$\mathbf{x} \;\alpha\; \alpha_1 \qquad \Lambda(\mathbf{x}) = \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} \geq \frac{P(\omega_2)}{P(\omega_1)}$$

☞ If $\lambda_{21}=\lambda_{12}$, minimum risk and MAP criteria are equivalent.

Discriminant Functions

- A useful way of representing classifiers is through discriminant functions $g_i(\mathbf{x})$ (i = 1, 2, ..., C), where the classifier assigns a feature vector $\mathbf{x}$ to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \forall\ j \neq i$$

- These functions divide the feature space into C decision regions ($\Re_1, \ldots, \Re_C$), separated by decision boundaries.
- For the classifier that minimizes conditional risk

$$g_i(\mathbf{x}) = -R(\alpha_i \mid \mathbf{x})$$

- For the classifier that minimizes error

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x})$$

Nota: Results do not change even if we replace every $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$ where $f(\cdot)$ is a monotonically increasing function (e.g., logarithm).

Discriminant Functions

- In the following, we will study in detail the behavior of the minimum-error-rate discriminant functions for classification problems characterized by C classes with multivariate Gaussian distributions:

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m_i})^t \Sigma_i^{-1}(\mathbf{x}-\mathbf{m_i})\right] = N(\mathbf{m_i}, \Sigma_i)$$

with $i = 1, \ldots, C$

- In this case, the discriminant function is written as

$$g_i(\mathbf{x}) = \ln\, p(\mathbf{x}\,|\,\omega_i) + \ln P(\omega_i)$$

$$\Downarrow$$

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mathbf{m_i})^t \Sigma_i^{-1}(\mathbf{x}-\mathbf{m_i}) - \frac{n}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

Case 1: $\Sigma_i = \sigma^2 I$

- Discriminant functions are

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + \mathrm{w}_{i0}$$

Linear discriminant

Threshold/bias

where

$$\begin{cases} \mathbf{w_i} = \dfrac{1}{\sigma^2}\mathbf{m_i} \\ \mathrm{w}_{i0} = -\dfrac{1}{2\sigma^2}\mathbf{m_i^t}\mathbf{m_i} + \ln P(\omega_i) \end{cases}$$

- Decision boundaries are the hypersurfaces corresponding to

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

Case 1: $\Sigma_i = \sigma^2 I$

- In this case, they can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x_0})=0$$

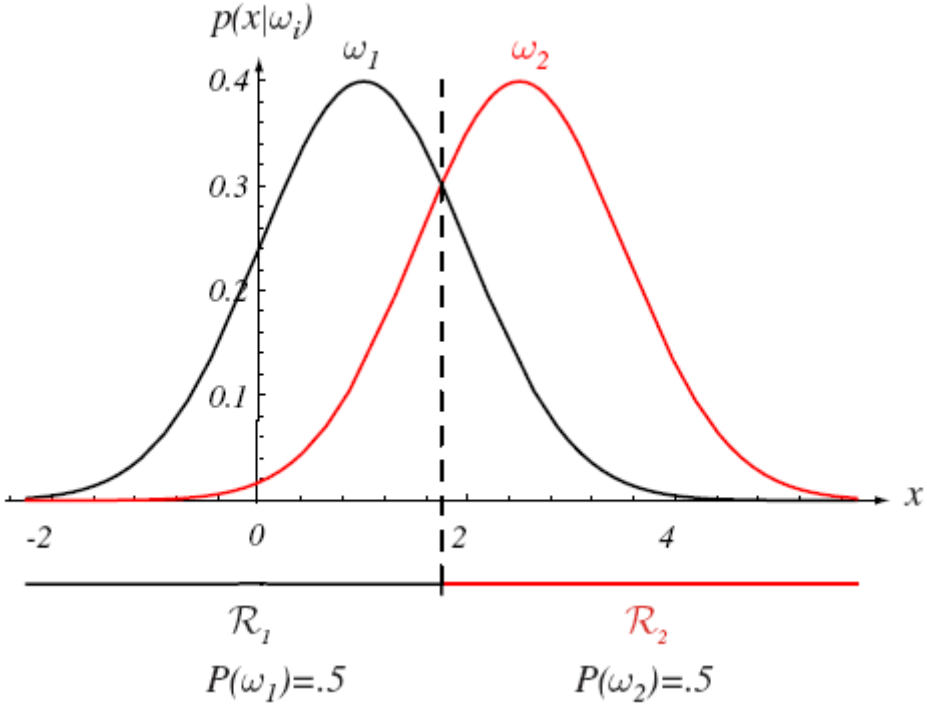
where

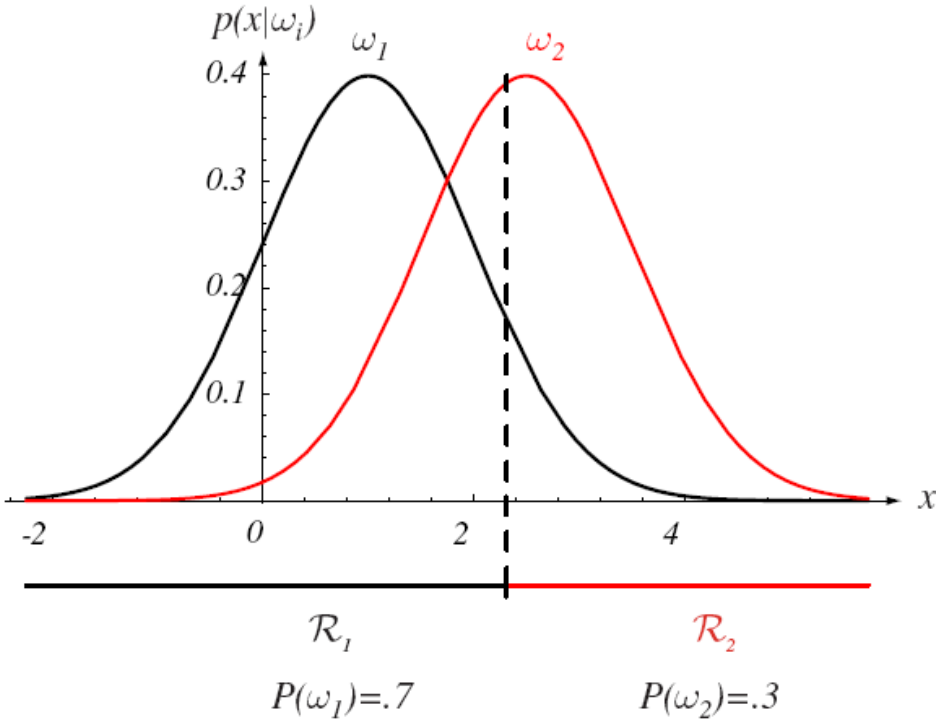
$$\begin{cases} \mathbf{w}=\mathbf{m_i}-\mathbf{m_j} \\ \mathbf{x_0}=\dfrac{1}{2}(\mathbf{m_i}+\mathbf{m_j})-\dfrac{\sigma^2}{\|\mathbf{m_i}-\mathbf{m_j}\|^2}\ln\dfrac{P(\omega_i)}{P(\omega_j)}(\mathbf{m_i}-\mathbf{m_j}) \end{cases}$$

- Hyperplane separating $\mathcal{R}_i$ and $\mathcal{R}_j$ passes through the point $\mathbf{x_0}$ and is orthogonal to the vector $\mathbf{w}$.

Case 1: $\Sigma_i = \sigma^2 I$

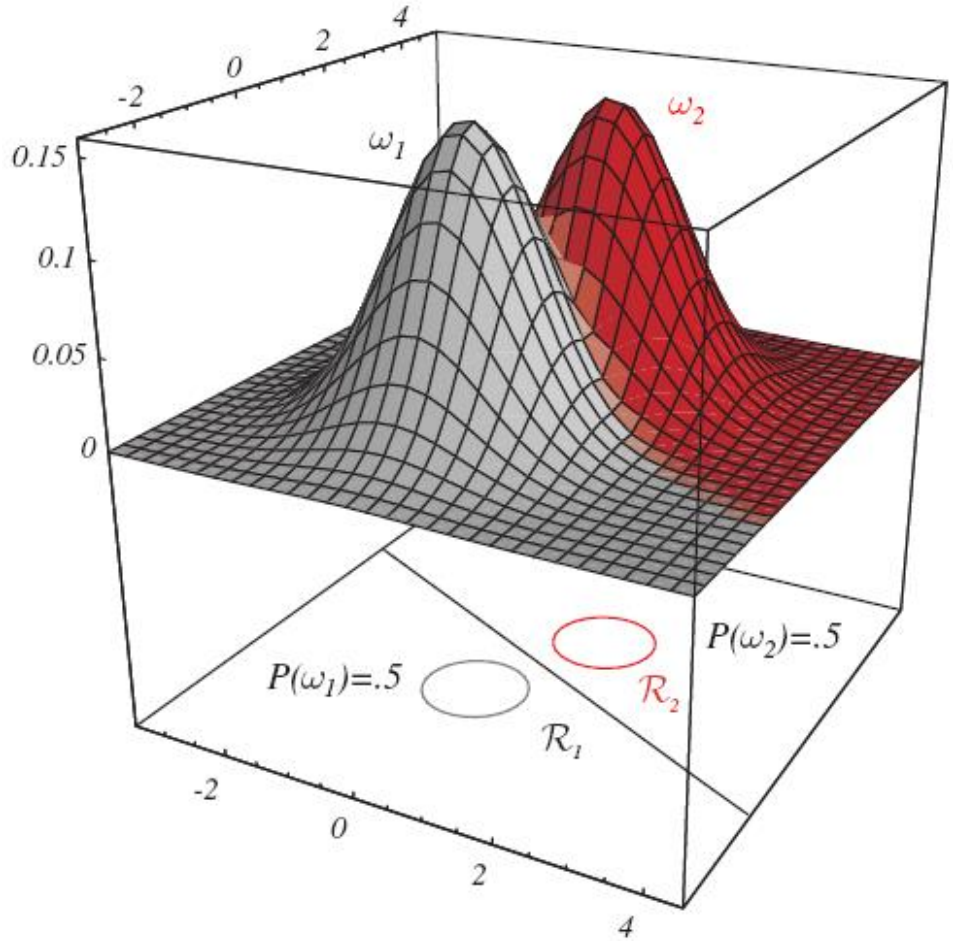
Examples with 1-D distributions

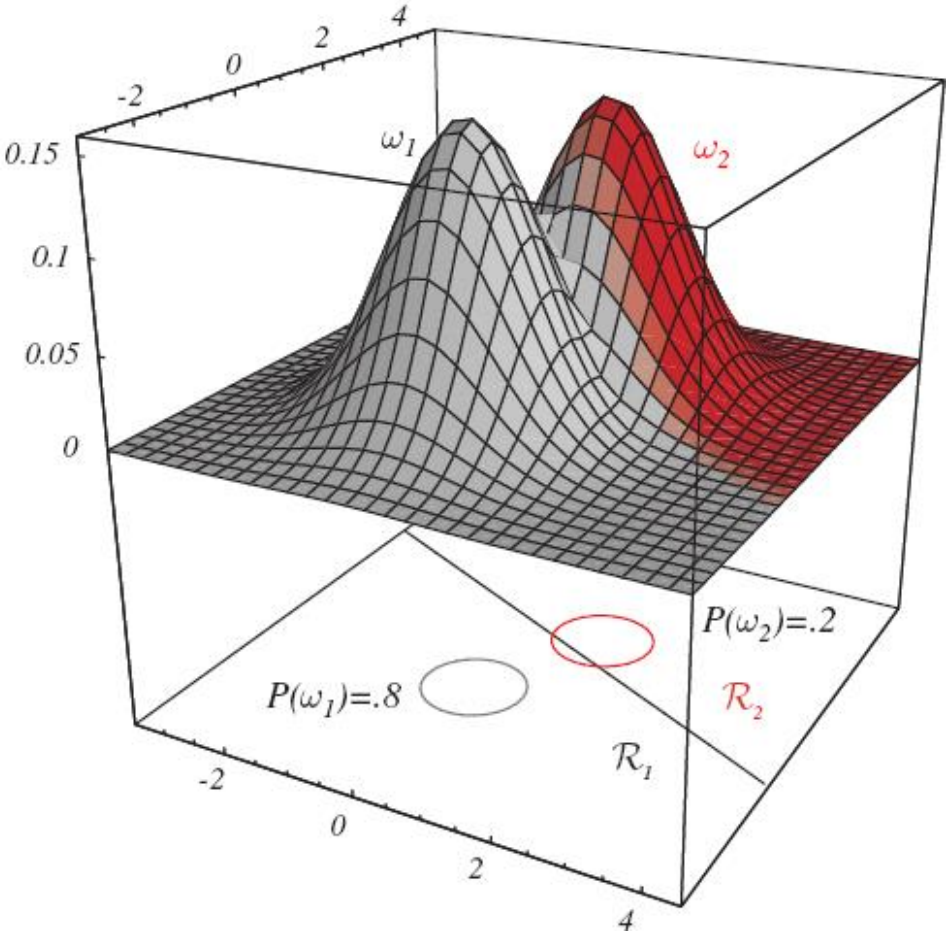




Case 1: $\Sigma_i = \sigma^2 I$

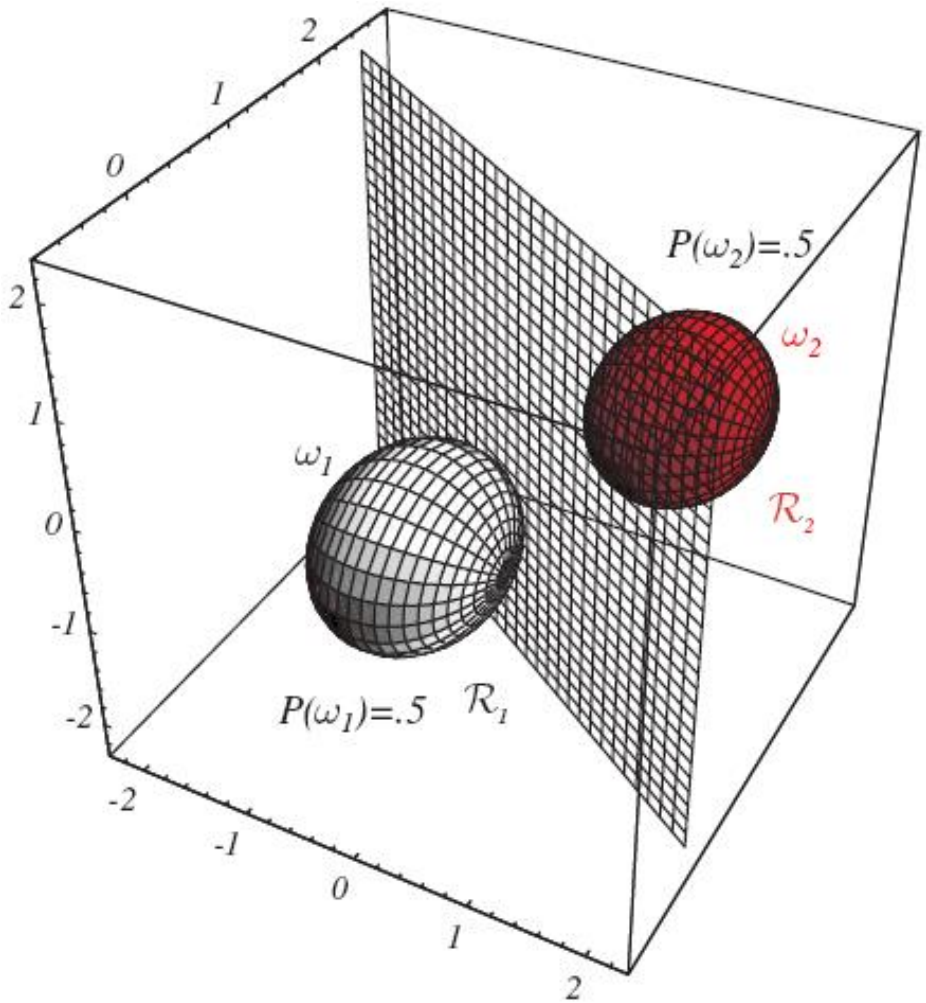
Examples with 2-D distributions

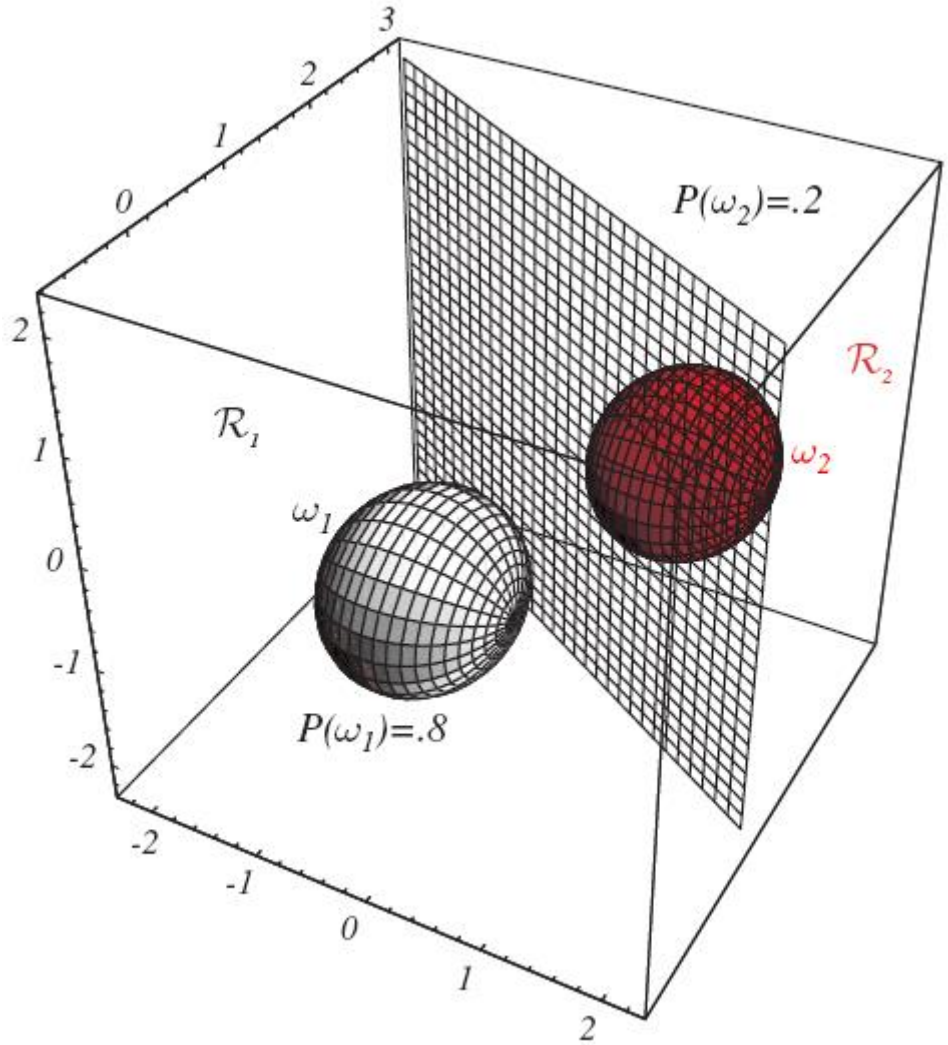




Case 1: $\Sigma_i = \sigma^2 I$

Examples with 3-D distributions





Case 2: $\Sigma_i = \Sigma$

- Discriminant functions become

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + \mathrm{w}_{i0}$$

Linear discriminant

where

$$\begin{cases} \mathbf{w_i} = \Sigma^{-1}\mathbf{m_i} \\ \mathrm{w}_{i0} = -\dfrac{1}{2}\mathbf{m_i^t}\Sigma^{-1}\mathbf{m_i} + \ln P(\omega_i) \end{cases}$$

Case 2: $\Sigma_i = \Sigma$

- Decision boundaries are

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x_0})=0$$

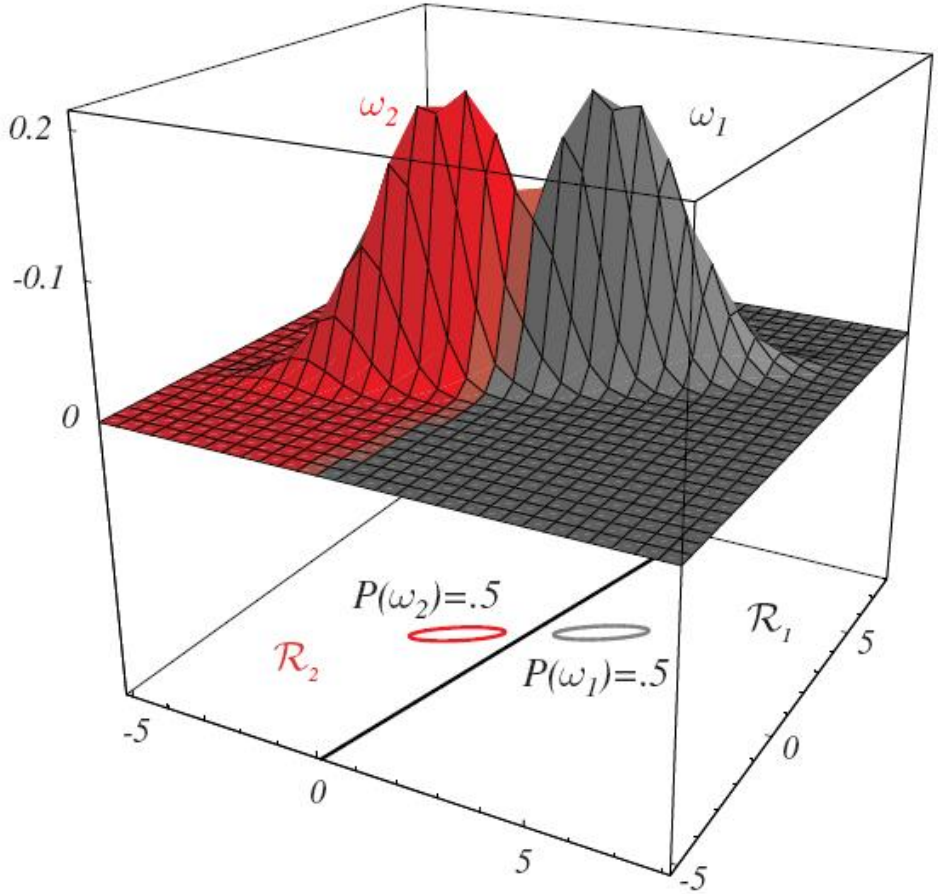
where

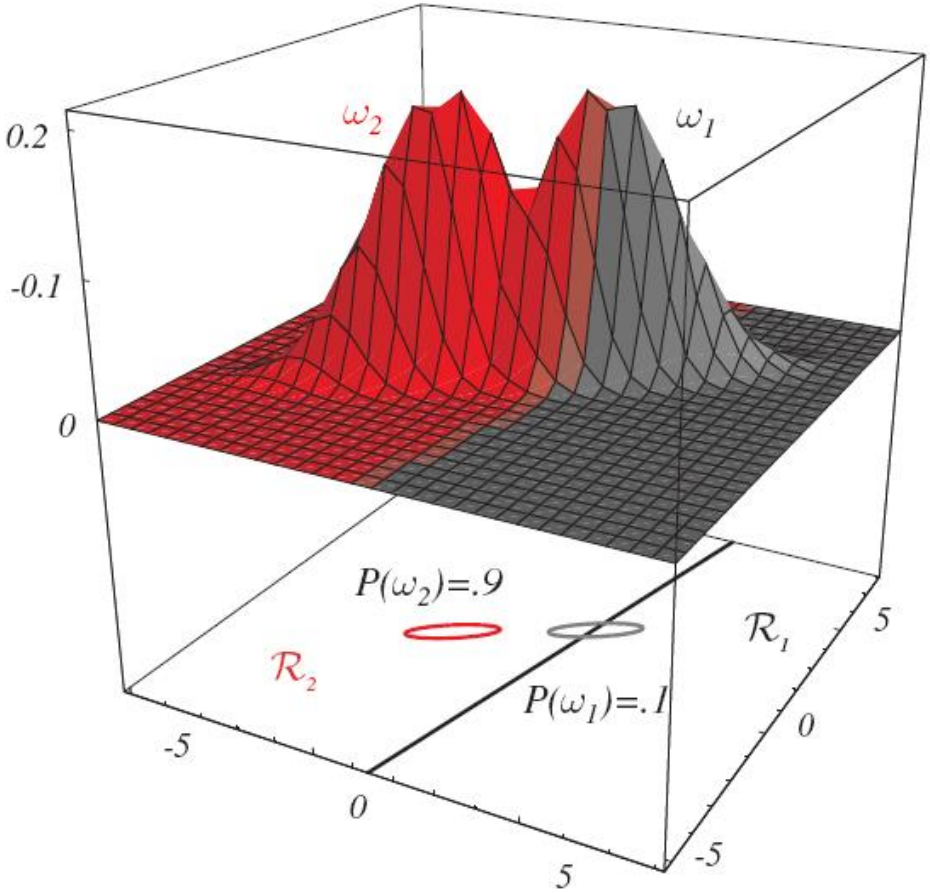
$$\begin{cases} \mathbf{w}=\Sigma^{-1}(\mathbf{m_i}-\mathbf{m_j}) \\ \mathbf{x_0}=\dfrac{1}{2}(\mathbf{m_i}+\mathbf{m_j})-\dfrac{\ln[P(\omega_i)/P(\omega_j)](\mathbf{m_i}-\mathbf{m_j})}{(\mathbf{m_i}-\mathbf{m_j})^t\Sigma^{-1}(\mathbf{m_i}-\mathbf{m_j})} \end{cases}$$

- Hyperplane passes through $\mathbf{x_0}$ but is not necessarily orthogonal to the line between the means.

Case 2: $\Sigma_i = \Sigma$

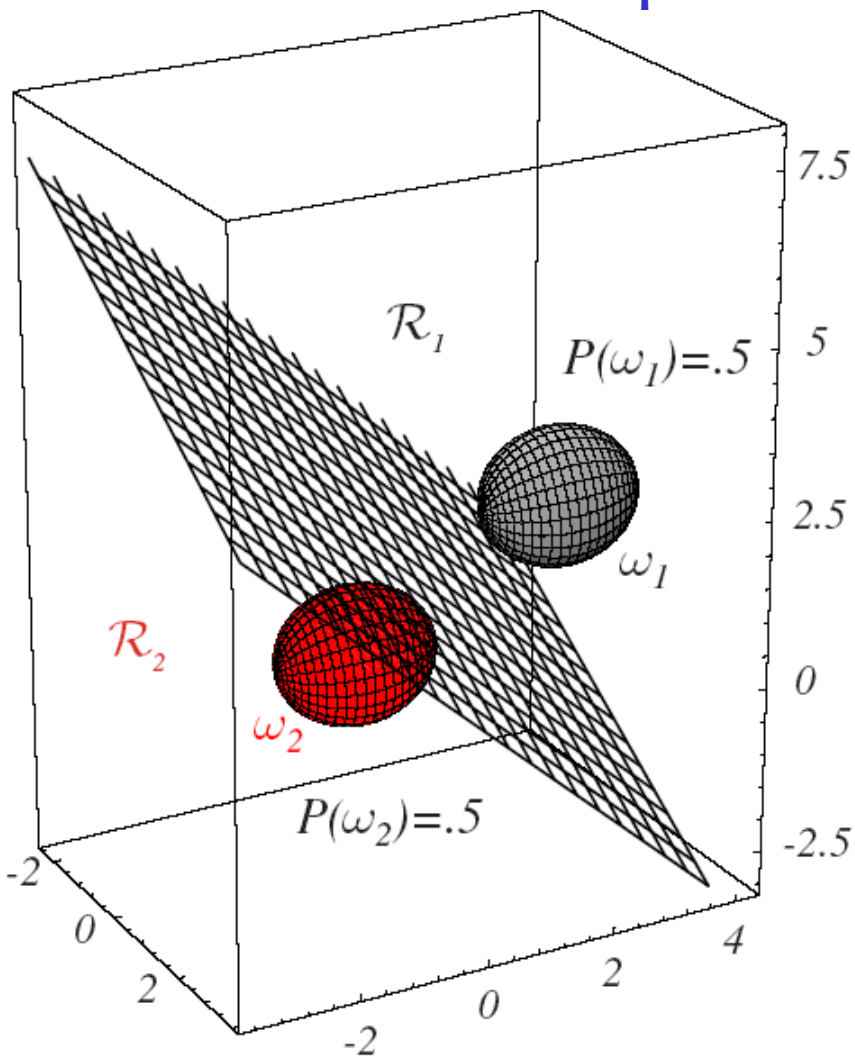
Examples with 2-D distributions

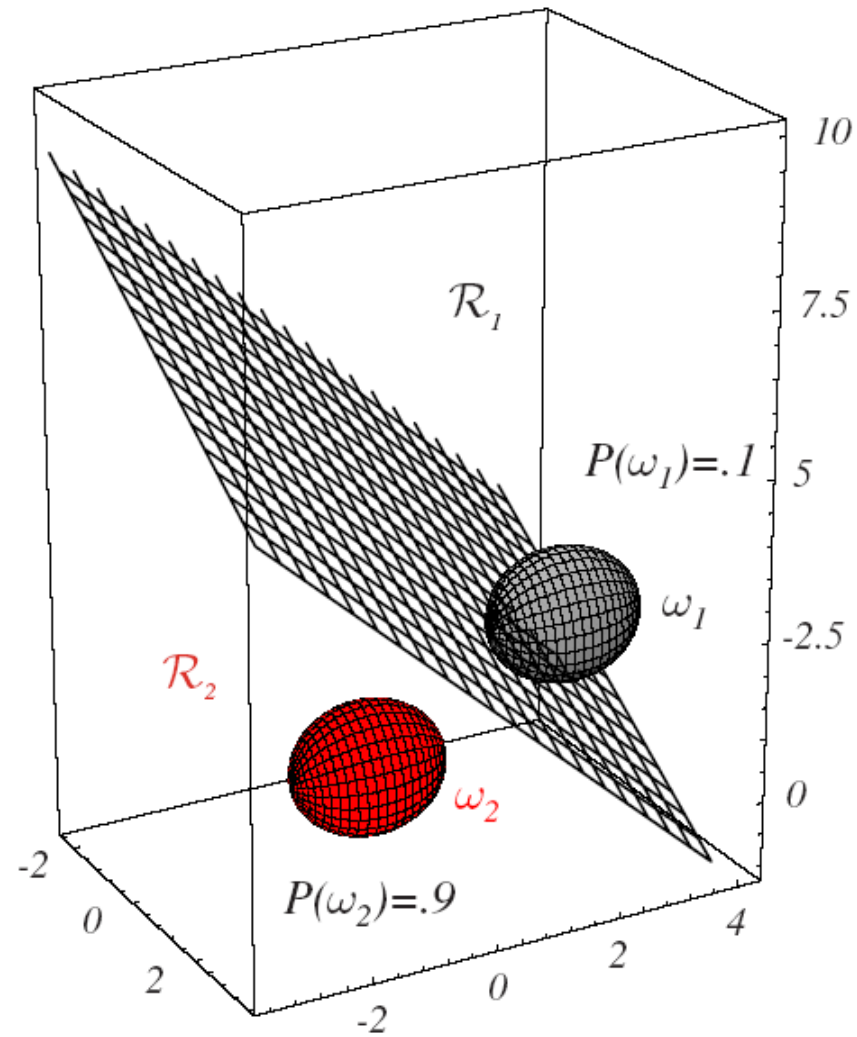




Case 2: $\Sigma_i = \Sigma$

Examples with 3-D distributions





Case 3: Σ_i = arbitrary

- Discriminant functions are

$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + \mathrm{w}_{i0}$$ Quadratic discriminant

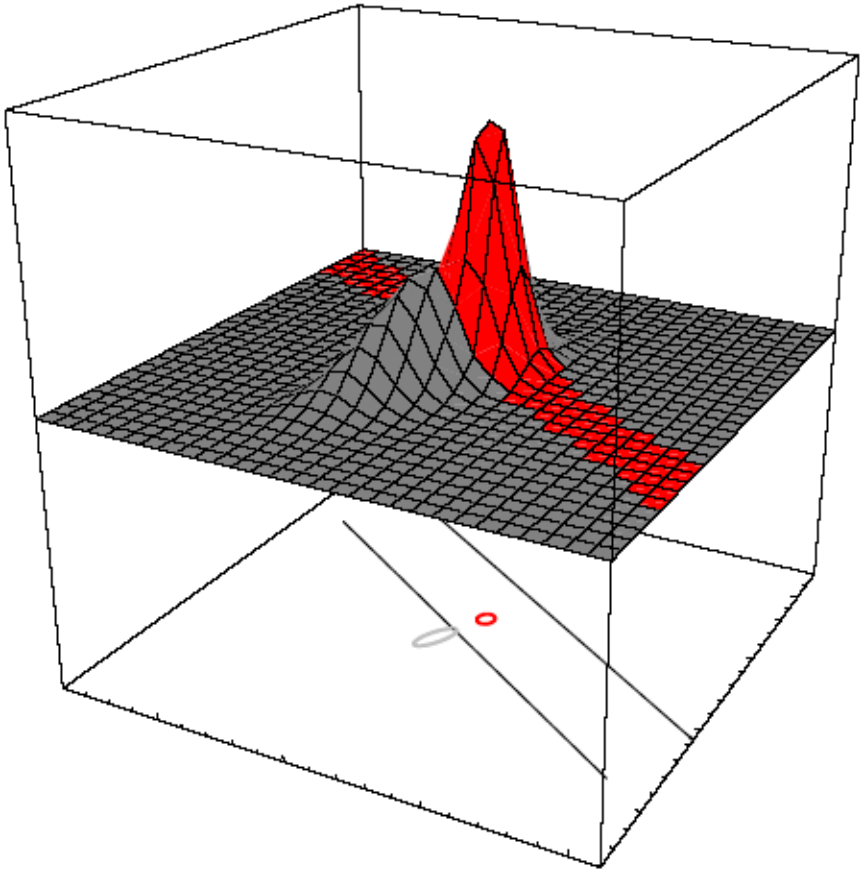
where

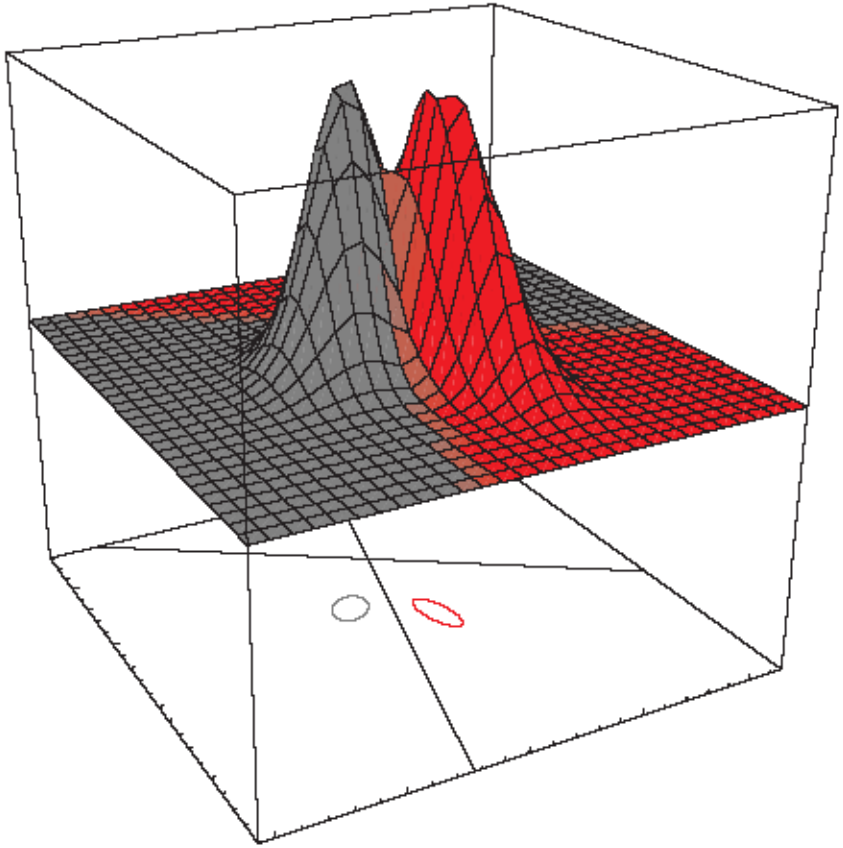
$$\begin{cases} \mathbf{W_i} = -\dfrac{1}{2}\Sigma_i^{-1} \\ \mathbf{w_i} = \Sigma_i^{-1}\mathbf{m_i} \\ \mathrm{w}_{i0} = -\dfrac{1}{2}\mathbf{m_i^t}\Sigma_i^{-1}\mathbf{m_i} - \dfrac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \end{cases}$$

- Decision boundaries are hyperquadrics.

Case 3: Σ_i = arbitrary

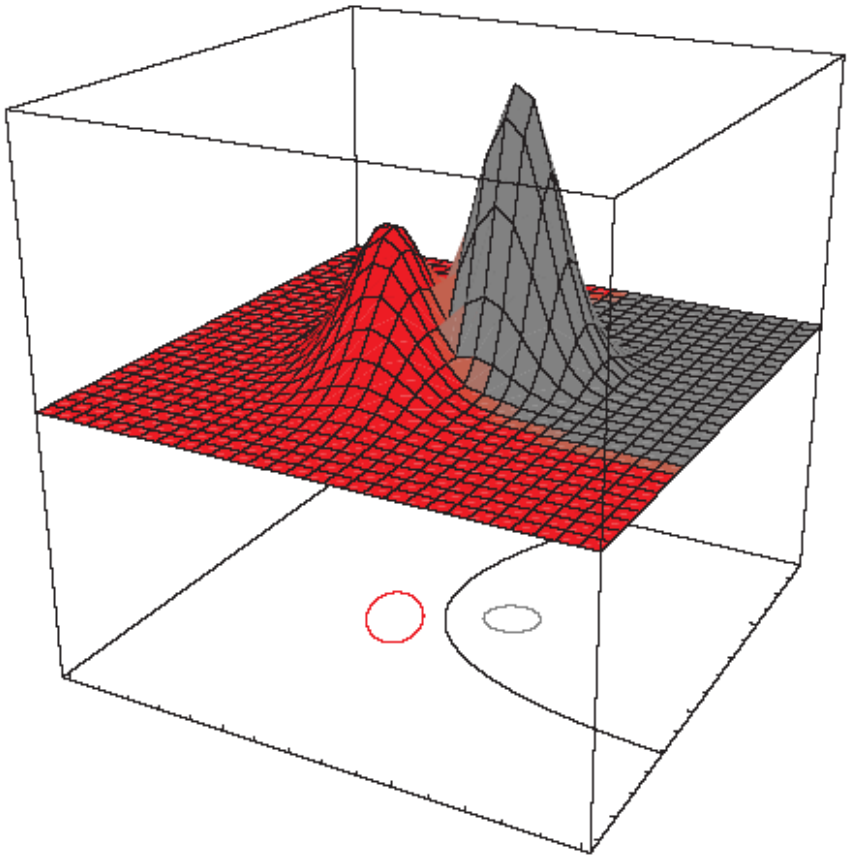
Examples with 2-D distributions

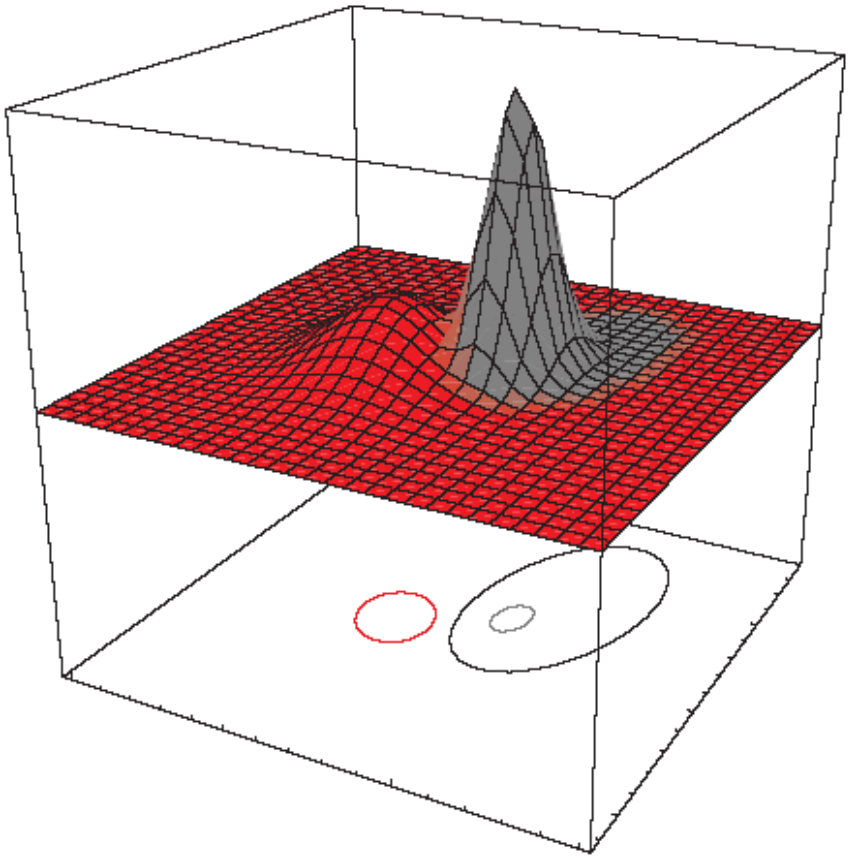




Case 3: Σ_i = arbitrary

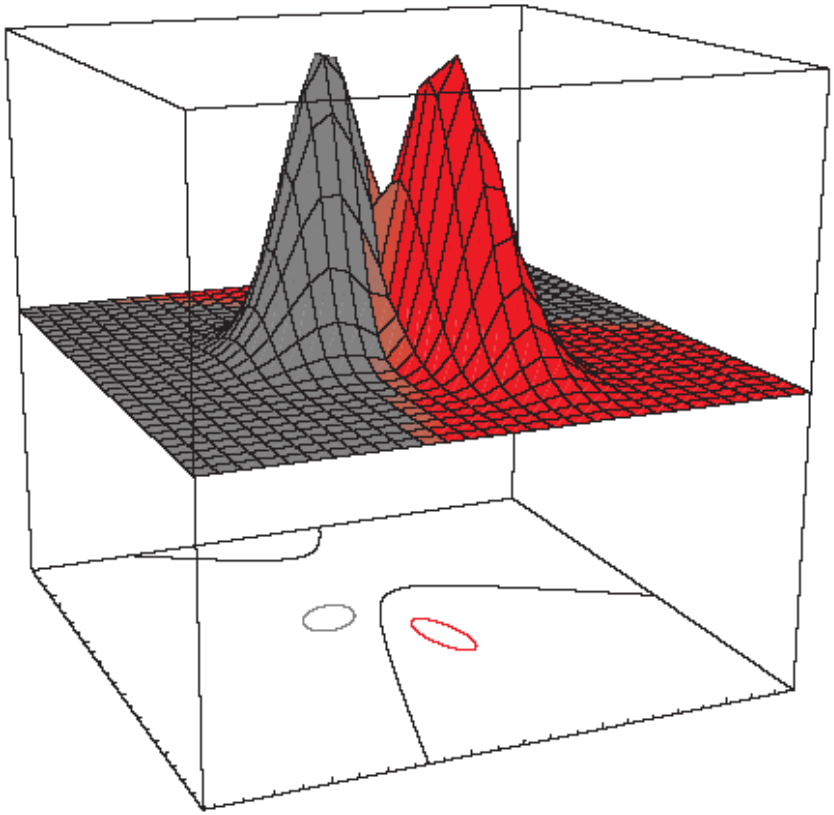
Examples with 2-D distributions

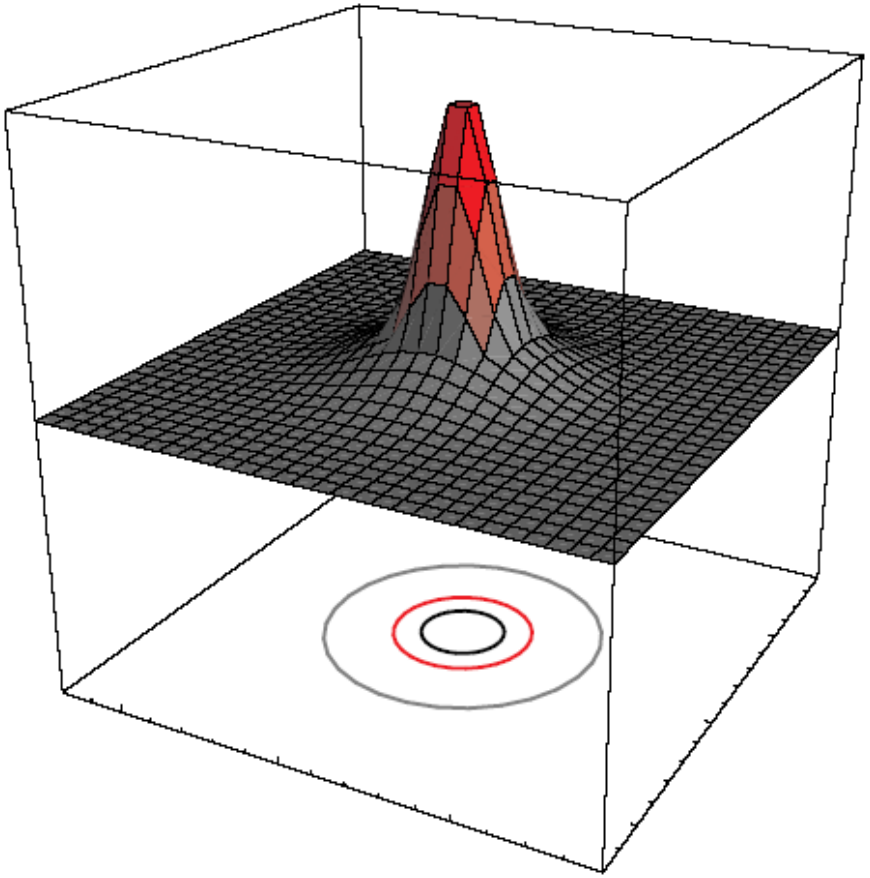




Case 3: Σ_i = arbitrary

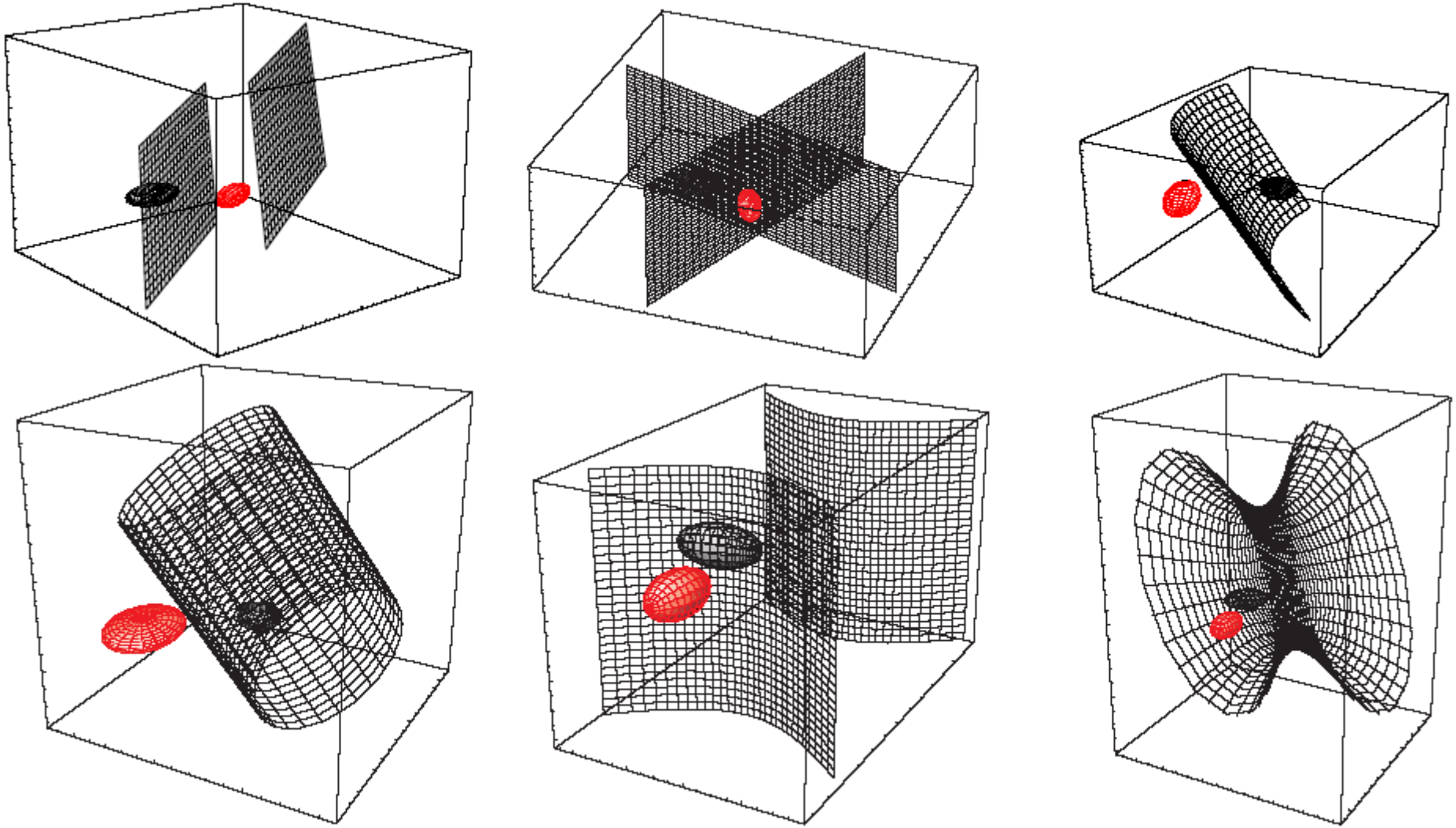
Examples with 2-D distributions





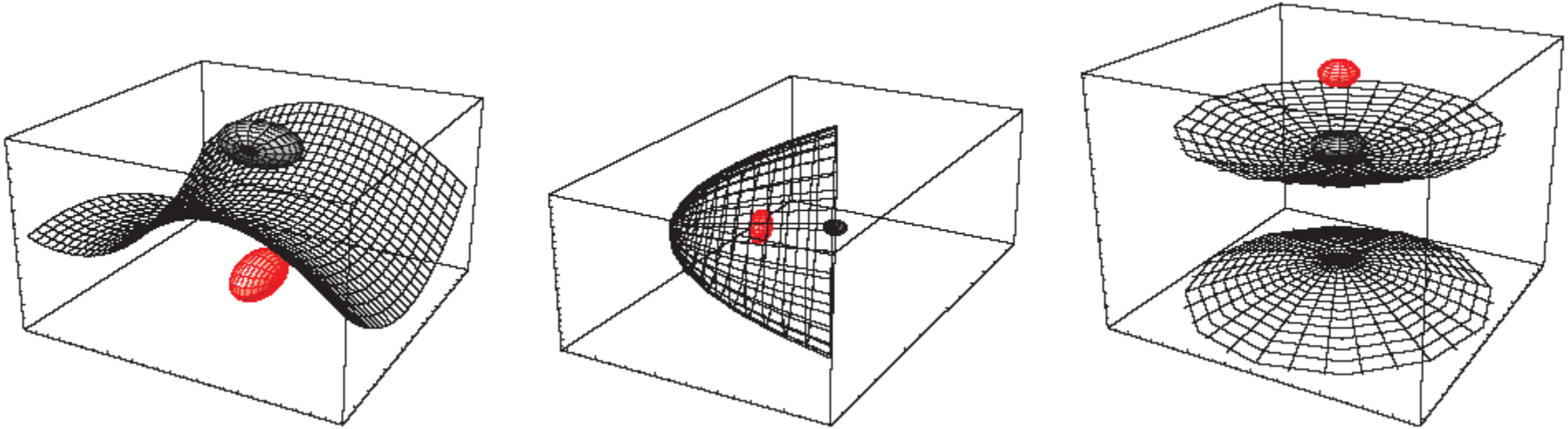
Case 3: Σ_i = arbitrary

Examples with 3-D distributions



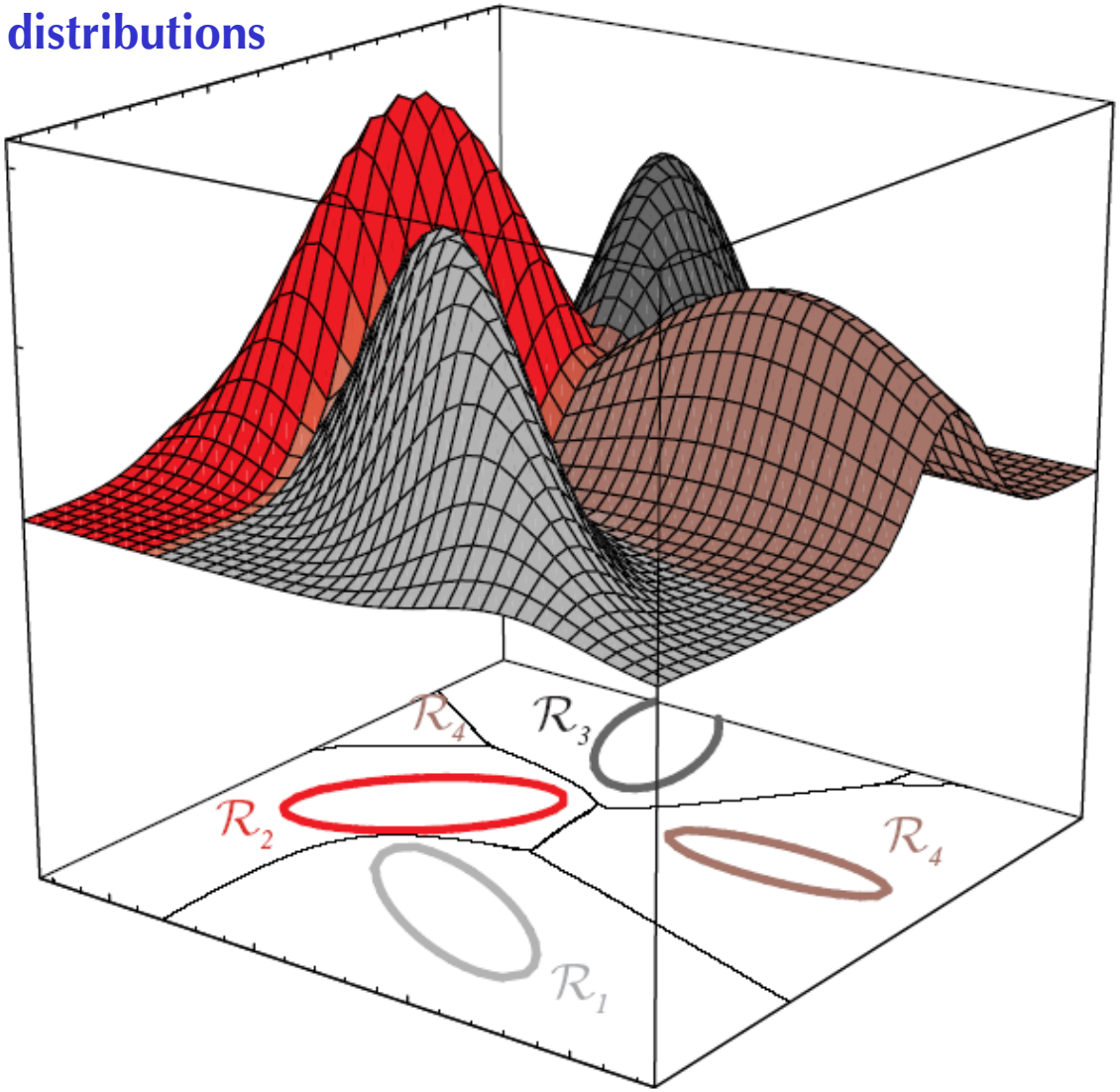
Case 3: Σ_i = arbitrary

Examples with 3-D distributions



Case 3: Σ_i = arbitrary

Example with 3-D distributions



Discriminant Functions: Example

- Let ω_1 and ω_2 be two classes such that:
  - $p(\mathbf{x}|\,\omega_i) = \mathrm{N}(\mathbf{m}_i, \Sigma_i)$ $(i = 1, 2)$
  - and

$$\begin{cases} P(\omega_1) = P(\omega_2) \\ \mathbf{m}_1 = (8, -4, -4) \\ \mathbf{m}_2 = (-96, 80, 80) \\ \Sigma_1 = \Sigma_2 = \Sigma = \begin{bmatrix} 8 & -4 & -4 \\ -4 & 8 & 4 \\ -4 & 4 & 8 \end{bmatrix} \end{cases}$$

Discriminant Functions: Example

- Applying what seen, the related discriminant functions are:

$$\begin{cases} g_1(\mathbf{x}) = \mathbf{w}_1^t \mathbf{x} + w_{10} = 4x_1 - \frac{3}{2} \\ g_2(\mathbf{x}) = -4x_1 + 8x_2 + 8x_3 - \frac{11}{2} \end{cases}$$

- The decision boundary takes thus the following form:

$$f_{12}(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = 8x_1 - 8x_2 - 8x_3 + 4 = 0$$

Decision Trees

- Most practical methods address problems, where feature vectors are real-valued and there exists some notion of metric.
- But suppose a classification problem involves nominal data, e.g., descriptions that are discrete and without any natural notion of similarity or even ordering.
- We will turn away from describing patterns by vectors of real numbers and toward using lists of attributes.
- A common approach is to specify the values of a fixed number of properties by a property *d*-tuple.

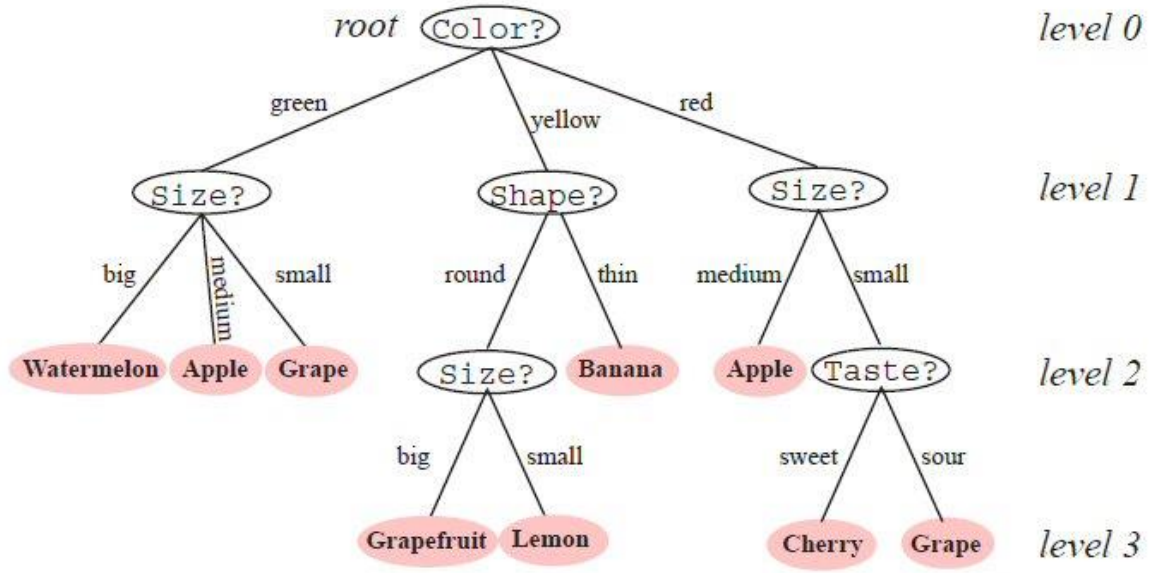
color = red
texture = shiny
taste = sweet
size = small

Decision Trees

- How can we best use such nominal data for classification?
- Most importantly, how can we efficiently learn categories using such non-metric data?
- In considering such problems, we move beyond the notion of continuous probability distributions and metrics toward discrete problems that are addressed by rule-based or syntactic pattern recognition methods.
- It is natural and intuitive to classify a pattern through a sequence of questions, in which the next question asked depends on the answer to the current question.

Decision Trees

- Such a sequence of questions is displayed in a directed decision tree or simply tree, where by convention the first or root node is displayed at the top, connected by successive (directional) links or branches to other nodes.
- These are similarly connected until link branch we reach terminal or leaf nodes, which have no further links.

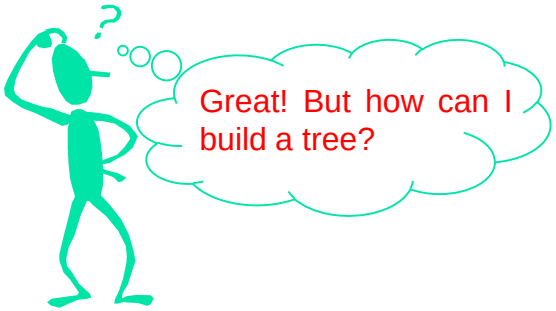


links must be mutually distinct and exhaustive

Leaf nodes bear a category labels

Decision Trees

- Decision trees bring several interesting properties:
  - Interpretability
  - Fast classification
  - Easy incorporation of prior knowledge from human experts

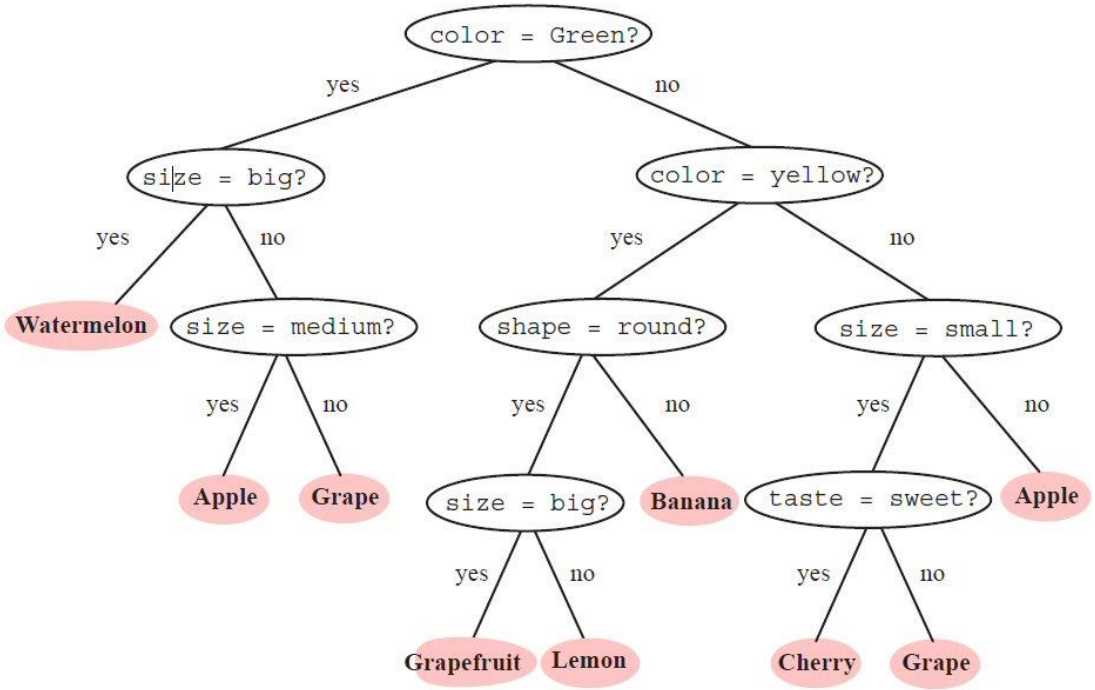


CART

- Classification and Regression Trees (CART) provide a general framework that can be instantiated in various ways to produce different decision trees.
- CARTs raise different main aspects which are:
  - Number of splits
  - Test selection and node impurity
  - When to stop splitting
  - Pruning
  - Assignment of leaf node labels

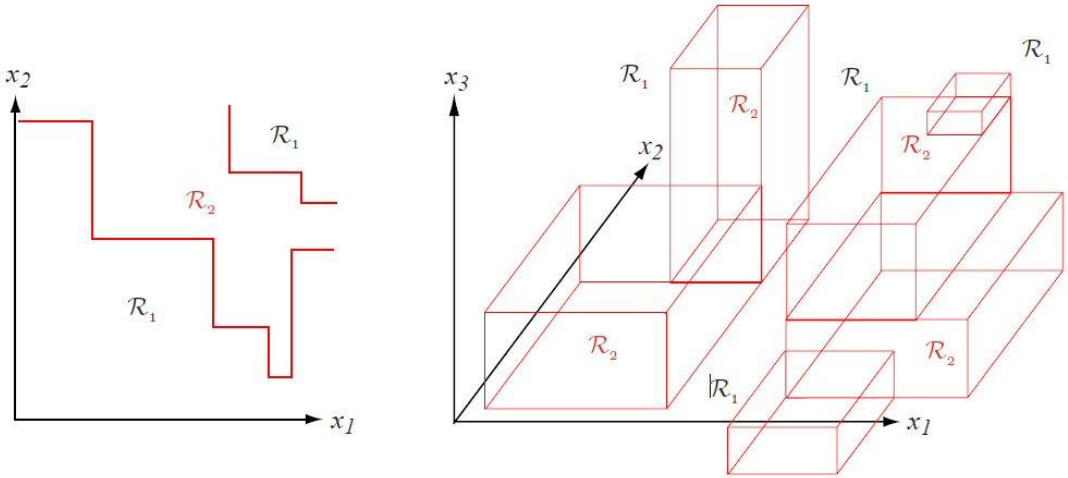
CART: Split Number

- Each decision outcome at a node is called a split, since it corresponds to splitting a subset of the training data.
- The root node splits the full training set. Each successive decision splits a proper subset of the data.



CART: Test Selection

- Much of the work in designing trees focuses on deciding which property test or query should be performed at each node.
- This is pretty important as it will define the decision boundaries produced by the tree.



Nota: We will focus on monothetic trees.

- The fundamental principle underlying tree creation is that of simplicity: we prefer decisions that lead to a simple, compact tree with few nodes.

CART: Node Impurity

- To this end, we seek a property test T at each node N that makes the data reaching the immediate descendent nodes as pure as possible.
- In formalizing this notion, it turns out to be more convenient to define the impurity, rather than the purity of a node.
- Several different mathematical measures of impurity have been proposed, all of which have basically the same behavior.
- Let $i(N)$ denote the impurity of a node N.
- In all cases, we want $i(N)$ to be 0 if all of the patterns that reach the node bear the same category label, and to be large if the categories are equally represented.

CART: Node Impurity

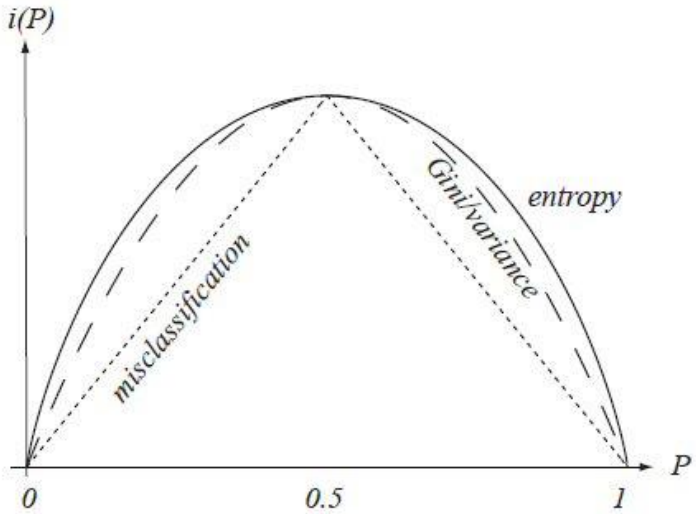
- Entropy impurity: $i(N) = -\sum_j P(\omega_j) log_2 \left(P(\omega_j)\right)$

  Fraction of patterns at node N belonging to ω_j

- Variance impurity: $i(N) = P(\omega_1)P(\omega_2)$

- Gini impurity: $i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = 1 - \sum_j P^2(\omega_j)$

- Misclassification impurity: $i(N) = 1 - \max P(\omega_j)$



CART: Node Impurity

- We now come to the key question: given a partial tree down to node *N*, what value *s* should we choose for the property test *T*?
- An obvious heuristic is to choose the test that decreases the impurity as much as possible. The drop in impurity is defined by:

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

Left descendent node

Right descendent node

Nota: For multibranch trees, suitable drop versions have to be used.

- The best test value *s* is the choice for *T* that maximizes $\Delta i(T)$.
- The optimization is local (done at a single node).
- There is no guarantee that successive locally greedy optimal decisions lead to the *global* optimum

CART: Twoing Criterion

- In multiclass binary tree creation, the twoing criterion may be useful.
- The overall goal is to find the split that best splits groups of the C categories, i.e., a candidate super-category C_1 consisting of all patterns in some subset of the categories, and candidate super-category C_2 as all remaining patterns.
- For every candidate split s, we compute a change in impurity $\Delta i(s;C_1)$ as though it corresponded to a standard two-class problem.
- That is, we find the split $\hat{s}(C_1)$ that maximizes the change in impurity.
- Finally, we find the super-category $\hat{C}_1$ which maximizes $\Delta i(\hat{s}(C_1); C_1)$.

CART: Stopped Splitting

- If we continue to grow the tree fully until each leaf node corresponds to the lowest impurity, then the data has typically been overfit.
- Conversely, if splitting is stopped too early, then the error on the training data is not sufficiently low and hence performance may suffer.
- How shall we decide when to stop splitting?
  - Cross-validation
  - Impurity reduction threshold
  - Complexity-accuracy tradeoff criterion
  - Hypothesis testing

CART: Pruning

- In stopped splitting (called also pre-pruning), node *N* might be declared a leaf, cutting off the possibility of beneficial splits in subsequent nodes.
- Accordingly, a stopping condition may be met too early for overall optimal recognition accuracy.
- The principal alternative approach to stopped splitting is pruning.
- In pruning, a tree is grown fully, that is, until leaf nodes have minimum impurity. Then, all pairs of neighboring leaf nodes (i.e., ones linked to a common antecedent node, one level above) are considered for elimination.
- Any pair whose elimination yields a satisfactory (small) increase in impurity is eliminated, and the common antecedent node declared a leaf.

CART: Pruning

- Clearly, such merging or joining of the two leaf nodes is the inverse of splitting.
- It is not unusual that after such pruning, the leaf nodes lie in a wide range of levels and the tree is unbalanced.
- The main benefit of pruning (called also post-pruning) is it directly uses all information in the training set.
- Naturally, this comes at a greater computational expense than stopped splitting. For problems with large training sets, the expense can be prohibitive.

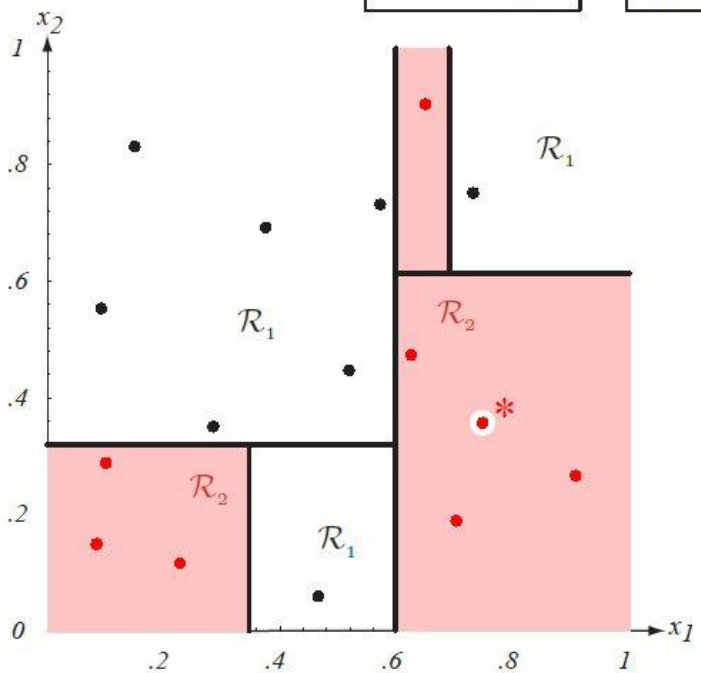
CART: Leaf Node Label Assignment

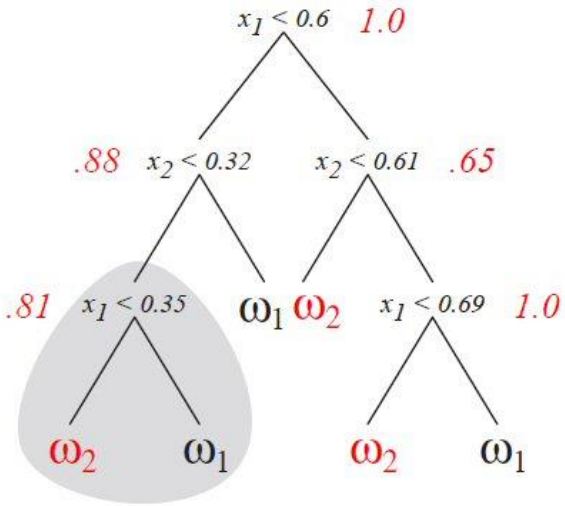
- Assigning category labels to the leaf nodes is the simplest step in tree construction.
- If successive nodes are split as far as possible, and each leaf node corresponds to patterns in a single category (zero impurity), then of course this category label is assigned to the leaf. Note that an extremely small impurity is not necessarily desirable, since it may be an indication that the tree is overfitting the training data.
- In the more typical case, where either stopped splitting or pruning is used and the leaf nodes have positive impurity, each leaf should be labeled by the category that has most points represented.

CART: Example

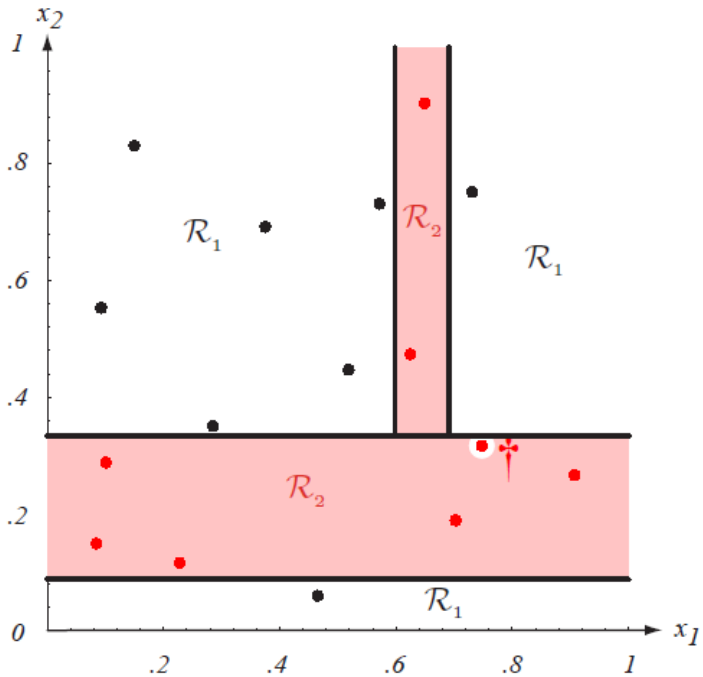
ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

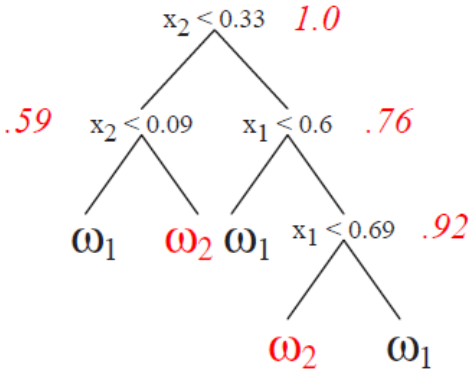
ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)





CART: Example



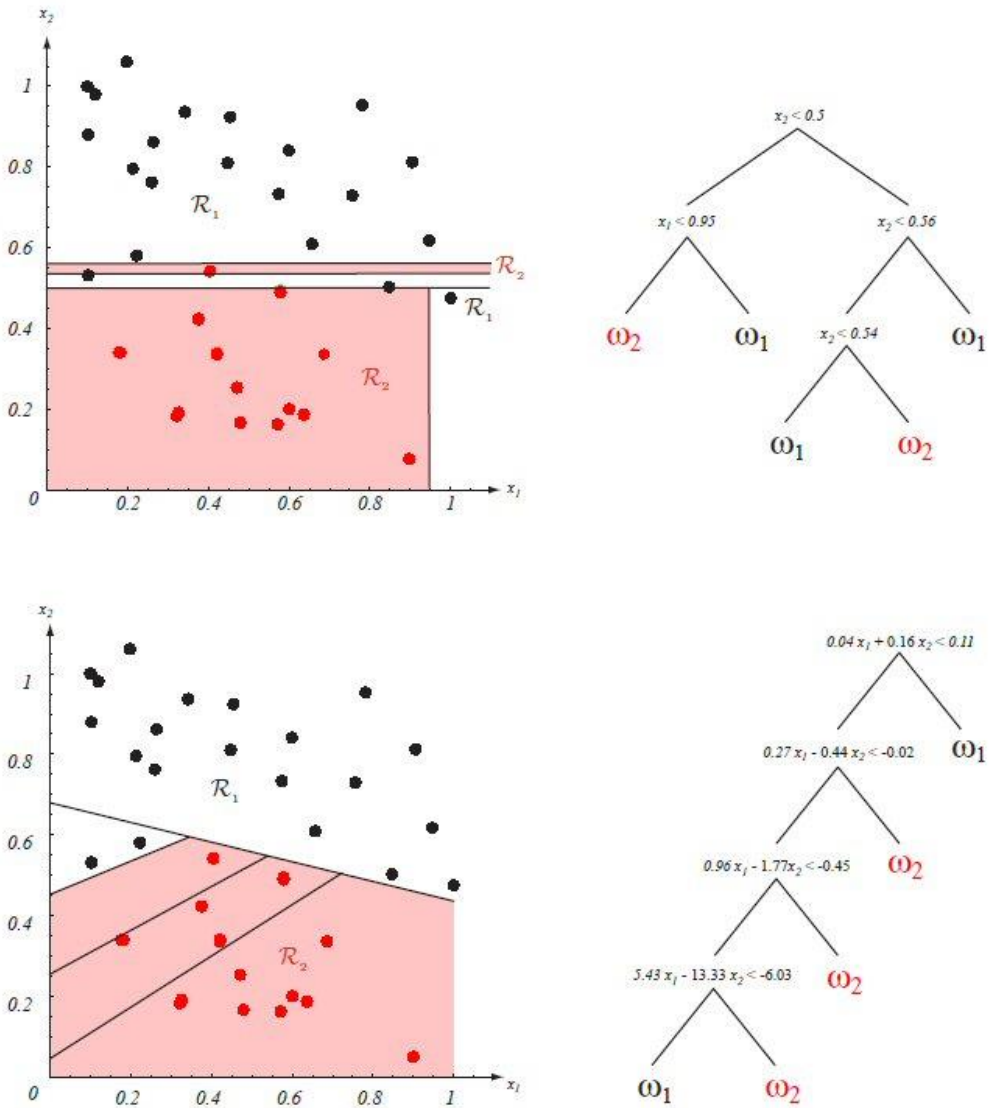


Nota: CARTs are subject to instability issue.

CART: Multivariate Decision Trees

- If the natural splits of real-valued data do not fall parallel to the feature axes or the full training data set differs significantly from simple or accommodating distributions, then the above method may lead to poor generalization.
- The simplest solution is to allow splits that are not parallel to the feature axes, such as a linear classifier trained via gradient descent on a classification or sum-squared-error criterion.

CART: Multivariate Decision Trees

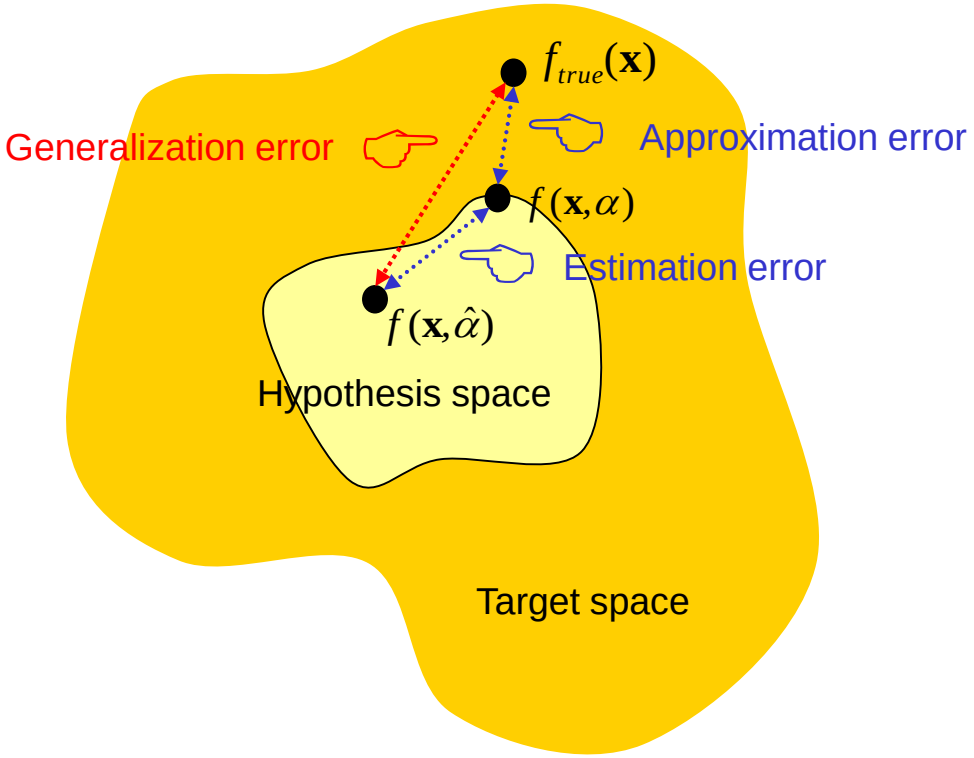


Other Tree Methods

- Virtually all tree-based classification techniques can incorporate the fundamental concepts described previously.
- While most tree-growing algorithms use an entropy impurity, there are many choices for stopping rules, for pruning methods and for the treatment of missing attributes.
- Two other popular tree algorithms are:
  - ID3
  - C4.5
- It is noteworthy that instability issues in decision trees gave birth to an interesting approach called random forest, which has been showing very effective.

Generalization Error

- The goal in modeling is to choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space.
- Errors in doing this arise from two cases:



Generalization Error

- Let us consider a machine whose task is to learn the mapping $\mathbf{x}_i \rightarrow y_i$, assuming that there exists some unknown probability distribution $\mathrm{P}(\mathbf{x},y)$ from which (training) data are drawn.
- The data are assumed independently drawn and identically distributed (i.i.d.).
- The machine is actually defined by a set of possible mappings $\mathbf{x} \rightarrow f(\mathbf{x},\alpha)$, where the functions $f(\mathbf{x},\alpha)$ themselves are labeled by the adjustable parameters α.
- The machine is assumed to be deterministic, i.e., for a given input $\mathbf{x}$, and choice of α, it will always give the same output $f(\mathbf{x},\alpha)$.
- A particular choice of α generates what we will call a trained machine.

Generalization Error

- The expectation of the generalization error for a trained machine is given by:

$$R(\alpha) = \int_{X \times Y} \underbrace{L\big(y, f(\mathbf{x}, \alpha)\big)}_{\text{Loss function, e.g., } L\big(y, f(\mathbf{x}, \alpha)\big) = \frac{1}{2}\left|y - f(\mathbf{x}, \alpha)\right|} p(\mathbf{x}, y)\, d\mathbf{x}\, dy$$

- This is a nice way of writing the true mean error, but unless we have an estimate of what $p(\mathbf{x},y)$ is, it is not very useful.
- The quantity $R(\alpha)$ is called the expected risk, or just the risk.

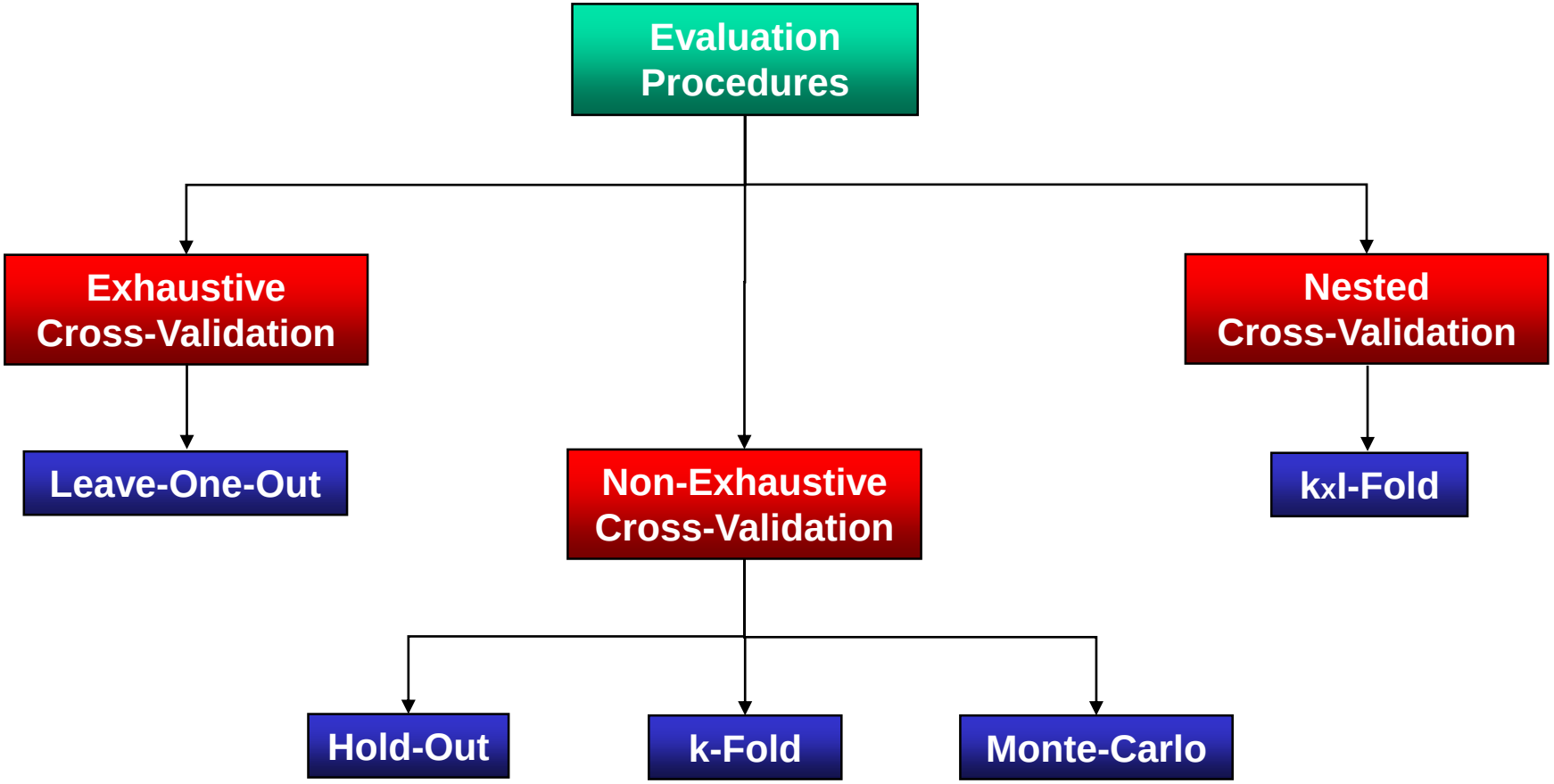
Accuracy Evaluation

- Empirical risk $R_{emp}(\alpha)$ is defined to be the measured average loss (e.g., mean error rate) on the training set (for a fixed, finite number of observations):

$$R_{emp}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, f(\mathbf{x}_i, \alpha)\big)$$

- Such an approximation of the generalization error is valid only if N>>.
- As in general N<<, we have but the choice to resort to some approximation to estimate the accuracy of a machine.
- In general, accuracy evaluation is important to:
  - Tune hyperparameters
  - Evaluate effectiveness of a trained machine
  - Make statistical comparison between different machines

Accuracy Evaluation

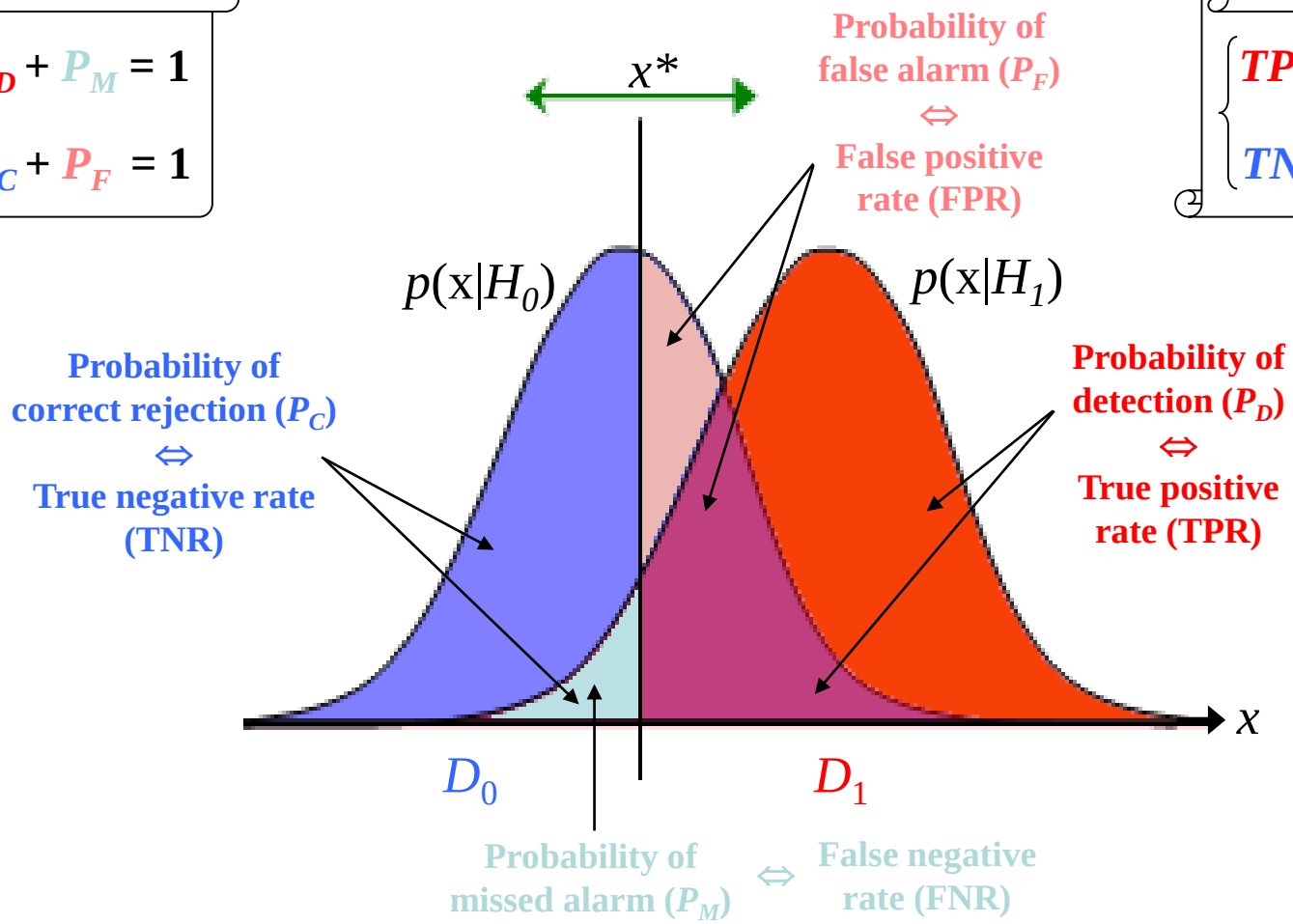


Binary Classifiers

Illustration of various probabilities for 1-D normal distributions

$$\begin{cases} P_D + P_M = 1 \\ P_C + P_F = 1 \end{cases}$$

$$\begin{cases} TPR + FNR = 1 \\ TNR + FPR = 1 \end{cases}$$



Binary Classifiers

Confusion Matrix and Common Derivations

	Decision (D_0)	Decision (D_1)
True (H_0)	TN	FP
True (H_1)	FN	TP

$$Accuracy = \frac{TN+TP}{TN+TP+FN+FP}$$

$$Specificity = \frac{TN}{TN+FP}$$

$$Precision = \frac{TP}{TP+FP}$$

$$F_\beta = \frac{(1+\beta^2)(Precision \cdot Sensitivity)}{\beta^2 Precision + Sensitivity}$$

$$Sensitivity = \frac{TP}{TP+FN}$$

$$F_1 = \frac{2 \cdot Precision \cdot Sensitivity}{Precision + Sensitivity}$$

Also called 'Recall'

Binary Classifiers: ROC Curve

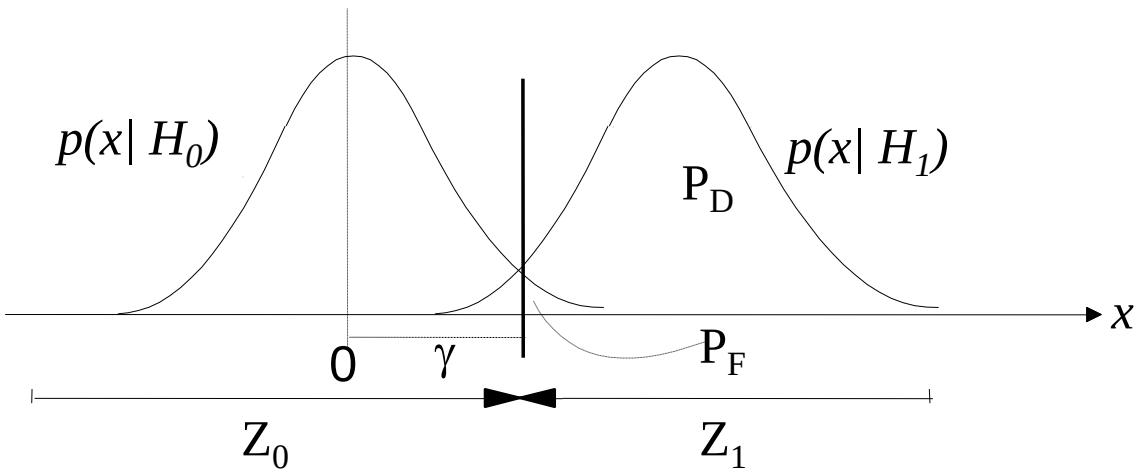
- The Receiver Operating Characteristic (ROC) is an important tool for assessing the performance of a binary classifier and thus for choosing the best decision strategy.

- The ROC curve captures the behavior of the TPR (P_D) as a function of the FPR (P_F), by varying the decision threshold η.

- It depends only on conditional pdfs $p(\mathbf{x}|H_0)$ and $p(\mathbf{x}|H_1)$. It depends neither on the cost matrix nor on the priors.

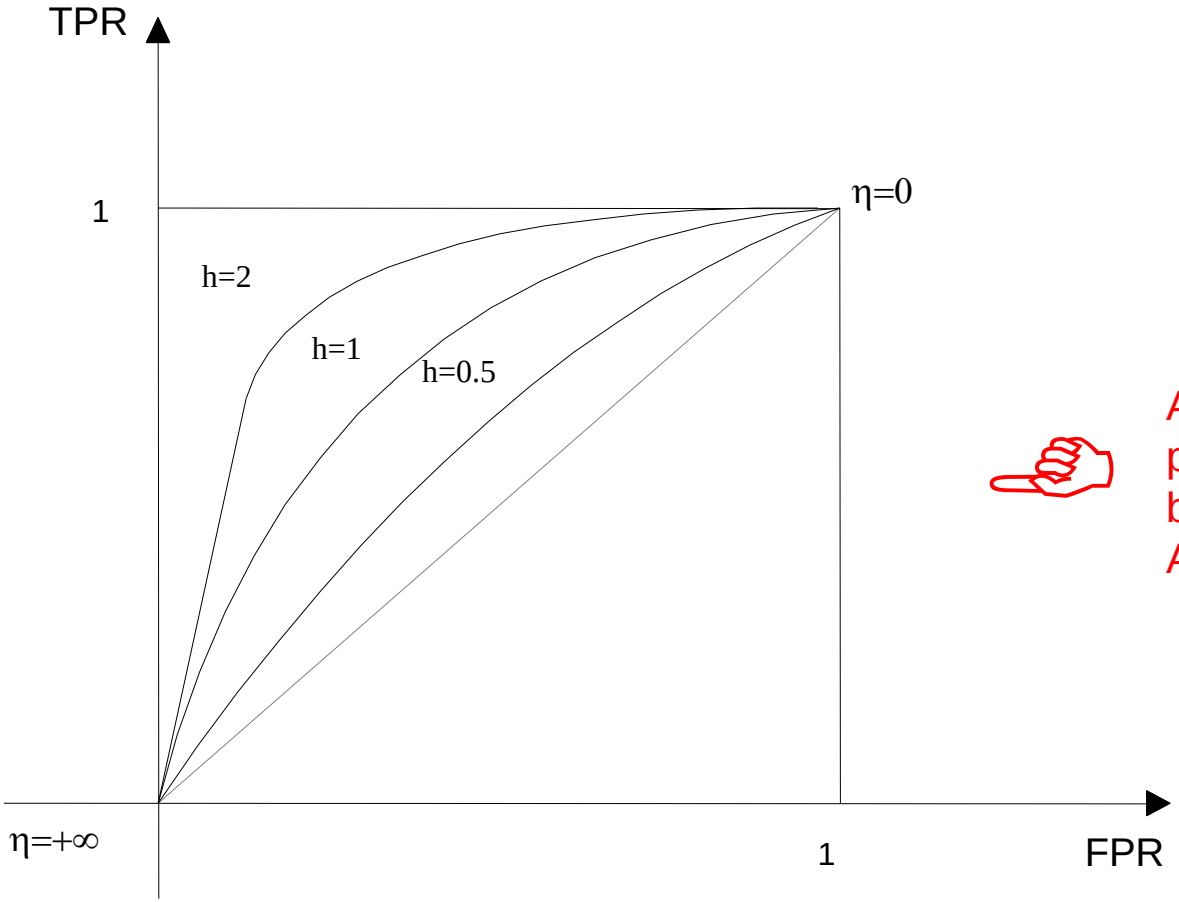
Binary Classifiers: ROC Curve

- Let us consider the following 1-D Gaussian case:
  - $n = 1,\ p(x|H_0) = N(0, \sigma^2),\ p(x|H_1) = N(m, \sigma^2)$



  - In this case, the ROC curves can be parameterized in terms of $h=m/\sigma= 0.5, 1, 2\ldots$

Binary Classifiers: ROC Curve



A single numeric value for performance evaluation could be extracted by computing the Area Under the Curve (AUC).

Multiclass Classifiers

Confusion Matrix and Common Derivations

	$\hat{\omega}_1$	$\hat{\omega}_2$	…	$\hat{\omega}_C$
ω_1	N_{11}	N_{12}	…	N_{1C}
ω_2	N_{21}	N_{22}	…	N_{2C}
…	…	…	…	…
ω_C	N_{C1}	N_{C2}	…	N_{CC}

$$Overall\ Accuracy = \frac{\sum_{i=1}^{C} N_{ii}}{\sum_{i=1}^{C}\sum_{j=1}^{C} N_{ij}}$$

$$Accuracy(\omega_i) = \frac{N_{ii}}{\sum_{j=1}^{C} N_{ij}}$$

$$Average\ Accuracy = \frac{\sum_{i=1}^{C} Accuracy(\omega_i)}{C}$$

Nota: Sensitivity, precision, specificity and F1 measures could be computed for each class as well.

Comparing Classifiers

- Let's assume we have two trained classifiers C_1 and C_2.
- We desire to assess if the difference of accuracy between the two classifiers is statistically significant.
- In the following, we will see two different hypothesis testing methods:
  - T-test (Student test)
  - McNemar's test
- A hypothesis H_0 called null hypothesis is tested against another hypothesis H1 called alternative.
- The objective is thus to know when the differences in H_0 are due to randomness or not.

Comparing Classifiers: T-Test

- We first evaluate the accuracy of C_1 and C_2 using k-fold cross-validation:

Fold	C_1	C_2	Δ
Fold 1	92.4	92.0	+0.4
Fold 2	91.8	90.6	+1.2
…	…	…	…
Fold i	A_{1i}	A_{2i}	Δ_i
…	…	…	…
Fold K	89.4	90.4	-1.0

- H_0 : mean difference is zero.
- We will consider a paired and two-tailed test, and assume Δ_i is normally distributed.

Comparing Classifiers: T-Test

- T-value: $T = \frac{signal}{noise} = \frac{difference\ between\ group\ means}{valiability\ of\ groups}$
- In our case:

$$\bar{\Delta} = \frac{1}{k}\sum_{i=1}^{k}(A_{1i} - A_{2i}) = \frac{1}{k}\sum_{i=1}^{k}\Delta_i \quad \text{and} \quad Var(\Delta) = \frac{1}{k-1}\sum_{i=1}^{k}(\Delta_i - \bar{\Delta})^2$$

$$T = \frac{\bar{\Delta}}{\sqrt{\frac{Var(\Delta)}{k}}}$$

Comparing Classifiers: T-Test

- In order to reject or not H_0, we need to compare the T-value with a critical value T_c.

$T \leq T_c$

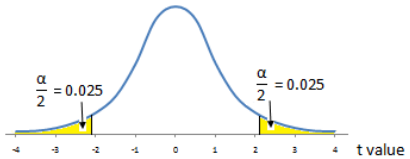
Accept H_0

Difference of accuracy between classifiers C_1 and C_2 is statistically insignificant.

Reject H_0

- T_c depends on the desired confidence level α (e.g., 5%) and degree of freedom df (df=k-1).

Student's t Distribution Table



	90%	95%	97.5%	99%	99.5%	99.95%	1-Tail Confidence Level
	80%	90%	95%	98%	99%	99.9%	2-Tail Confidence Level
	0.100	0.050	0.025	0.010	0.005	0.0005	1-Tail Alpha
df	0.20	0.10	0.05	0.02	0.01	0.001	2-Tail Alpha
1	3.0777	6.3138	12.7062	31.8205	63.6567	636.6192	
2	1.8856	2.9200	4.3027	6.9646	9.9248	31.5991	
3	1.6377	2.3534	3.1824	4.5407	5.8409	12.9240	
4	1.5332	2.1318	2.7764	3.7469	4.6041	8.6103	
5	1.4759	2.0150	2.5706	3.3649	4.0321	6.8688	
6	1.4398	1.9432	2.4469	3.1427	3.7074	5.9588	
7	1.4149	1.8946	2.3646	2.9980	3.4995	5.4079	
8	1.3968	1.8595	2.3060	2.8965	3.3554	5.0413	
9	1.3830	1.8331	2.2622	2.8214	3.2498	4.7809	
10	1.3722	1.8125	2.2281	2.7638	3.1693	4.5869	
11	1.3634	1.7959	2.2010	2.7181	3.1058	4.4370	
12	1.3562	1.7823	2.1788	2.6810	3.0545	4.3178	
13	1.3502	1.7709	2.1604	2.6503	3.0123	4.2208	
14	1.3450	1.7613	2.1448	2.6245	2.9768	4.1405	
15	1.3406	1.7531	2.1314	2.6025	2.9467	4.0728	
16	1.3368	1.7459	2.1199	2.5835	2.9208	4.0150	
17	1.3334	1.7396	2.1098	2.5669	2.8982	3.9651	
18	1.3304	1.7341	2.1009	2.5524	2.8784	3.9216	
19	1.3277	1.7291	2.0930	2.5395	2.8609	3.8834	
20	1.3253	1.7247	2.0860	2.5280	2.8453	3.8495	
21	1.3232	1.7207	2.0796	2.5176	2.8314	3.8193	
22	1.3212	1.7171	2.0739	2.5083	2.8188	3.7921	
23	1.3195	1.7139	2.0687	2.4999	2.8073	3.7676	
24	1.3178	1.7109	2.0639	2.4922	2.7969	3.7454	
25	1.3163	1.7081	2.0595	2.4851	2.7874	3.7251	
26	1.3150	1.7056	2.0555	2.4786	2.7787	3.7066	
27	1.3137	1.7033	2.0518	2.4727	2.7707	3.6896	
28	1.3125	1.7011	2.0484	2.4671	2.7633	3.6739	
29	1.3114	1.6991	2.0452	2.4620	2.7564	3.6594	
30	1.3104	1.6973	2.0423	2.4573	2.7500	3.6460	

Comparing Classifiers: McNemar's Test

- It is a paired nonparametric test, which aims at comparing two classifiers in a dataset after a hold-out process (on same test set).
- It operates upon a contingency table:

Sample	C_1	C_2
#1	correct	correct
#2	correct	wrong
…	…	…
#i	wrong	wrong
…	…	…
#N	wrong	correct

	C_2 correct	C_2 wrong
C_1 correct	N_{11}	N_{10}
C_1 wrong	N_{01}	N_{00}

- H_0 : C_1 and C_2 disagree in the same way.

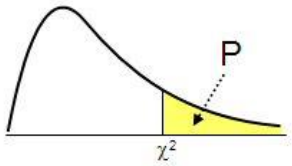
Comparing Classifiers: McNemar's Test

- The McNemar's test statistic is given by:

$$Z^2 = \frac{(N_{10} - N_{01})^2}{N_{10} + N_{01}}$$

- Under H_0 ($N_{01} \cong N_{10}$), Z^2 follows a χ^2 distribution with 1-degree of freedom. For a 95% confidence test, $\chi^2_{1,0.95}$=3.84.
- So if Z^2 is larger than 3.84, then with 95% confidence, we can reject the null hypothesis that the two classifiers behave in the same way.

Values of the Chi-squared distribution



	P										
DF	0.995	0.975	0.20	0.10	0.05	0.025	0.02	0.01	0.005	0.002	0.001
1	0.0000393	0.000982	1.642	2.706	3.841	5.024	5.412	6.635	7.879	9.550	10.828