Università degli studi di Trento

Machine Learning Module I

Prof. Melgani
Prof. Passerini

Academic Year 2024/2025

Contents

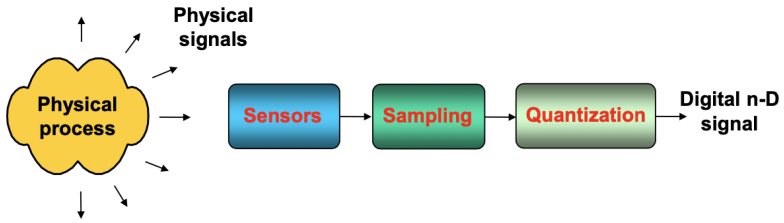
Chapter 1

Introduction

1.1 Mono/Multidimensional Signals

A signal is a varying quantity that carries information about physical phenomenon/process analysis. In a mono-dimensional signal a function f( ) represents the information evolution which respects to a variable which translates a physscal reality such as time, frequency, pressure, etc...
In a multidimensional signal information evolution is simultaneously related to multiple correlated or uncorrelated physical realities.

1.2 Signal acquisition



Sensors influence the quality of the signal, it can be affected by noise. Output of sensors are continuous signals in functions of time but it's not possible to analyze a continuous signal. We need to discretize the signal, by taking samples along the signal. This discretization in time is called *sampling*. A lot of samples offers more precision but on the other hand it costs a lot of memory (oversampling). Under sampling bring us to lose a lot of information from the signal. The correct number of samples (sampling frequency) is given by Nyquist criteria that allow to avoid overlap: $f_s = 2 \cdot f_{max}$. The process is: signal in time domain, Fourier transform that give a signal in frequency domain, calculate sampling frequency.

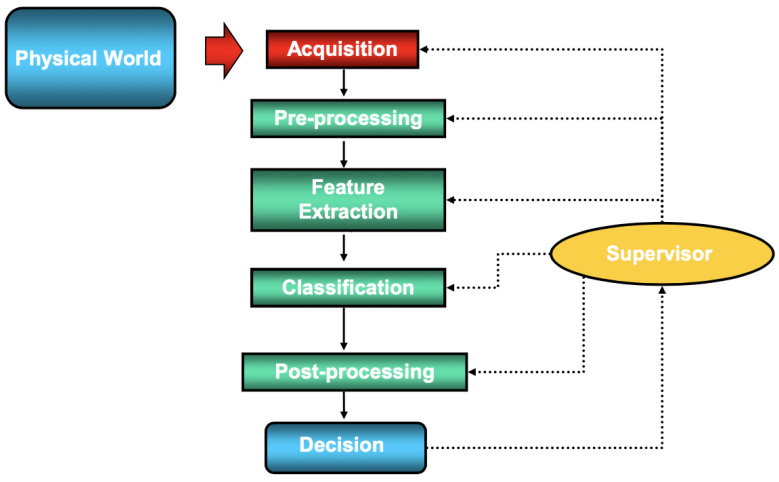
1.3 Learning

Machine learning is the study of computer algorithms that improve automatically through experience.

- Supervised learning: the learner is fed with a set of I/O pairs. We can have binary, multiclass (1 to 1 relation, 1 input - 1output)and multilabel classification (1 to N relation, 1 input - N outputs).
  Supervised Tasks: captioning (caption some details from an image. Output is discrete. Is like a sequential classifier), regression (quantity analysis. The output is continuous.) and ranking (e.g. priority=1. For selecting best features).
- Unsupervised learning: no output information is provided, but just input are modeled by the learner.
- Semi-supervised learning: besides a limited amount of I/O, the plenty unlabeled data are exploited during the learning process.
- Active learning: the learner interactively queries a supervisor to label new samples with desired outputs.
- Reinforcement learning: process of an arbitrary being (agent) in the world surrounding it (environment). The agent seeks to maximize the rewards it receives from the environment, and performs different actions in order to learn how the environment responds and gain more rewards.

1.4 ML System

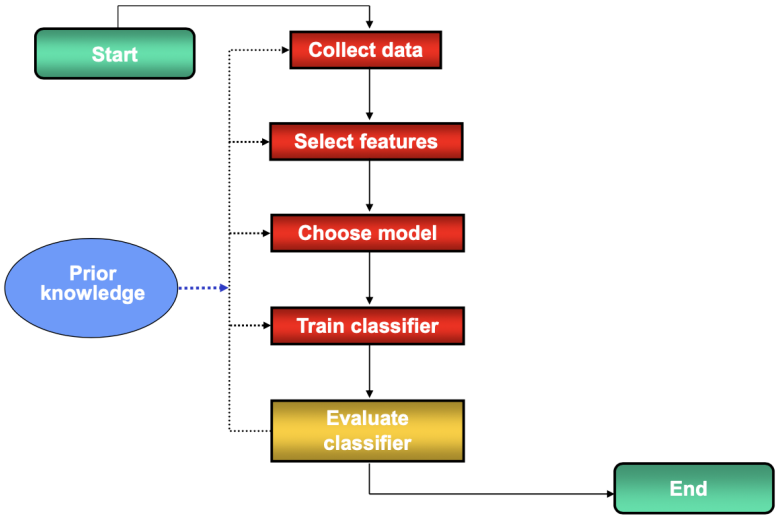
Block Scheme



In the acquisition process there can be one or more sensors, they provide raw signals. The quality of the sensor affects the signal (noise, distortion, precision,...). Before analyzing signals they have to be pre-processed to reduce noise. We want to stretch the histogram of the image for have a more contrast and readability of the image. There are 2 kinds of segmentation: high level (we have assign to each pixel a label (see multiclass classification) and low level (pixels that are very similar and contiguous -for example pixels near and of the same color- are grouped in a unique pixel/region). Features Extraction find a feature that characterize a pixel.

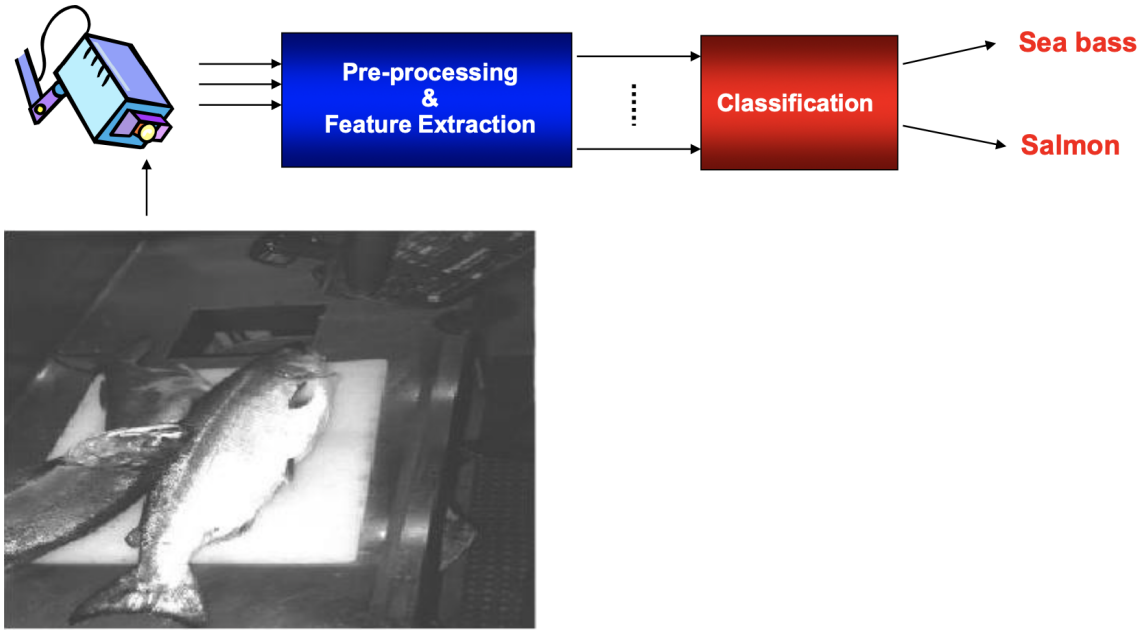
[R G B] of a picture can be the features vector. The major problem is to reduce the complexity of the classifier. Large model means a lot of weights to tune. To have less complexity we have to reduce the number of features as much as possible. To do this it's necessary to select the best features of our features set (Features Selection). Classification allow to move from the input space to the output space. The user set the number of classes for the classification. It depends on the number of features, training samples, noise (There may be an overlap of the classes). Post-processing is not a mandatory step but sometimes is useful.

Design Phases

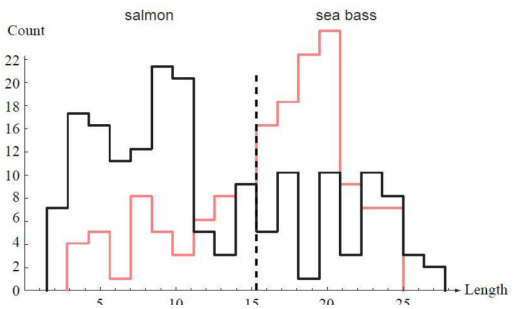


Collected data represents training samples. These training samples must be as much informative as possible. Features must be powerful for a discrimination capability and they should be easy and fast to extract and insensitive to scaling. The start is a simple model, then if it is not enough precise we can move to a more complex model. The objective is to have an output where the classes are perfectly separated. Train classifier means estimate the weights of the classifier. During the training phase we have to take in consideration two aspects: velocity and generalization.

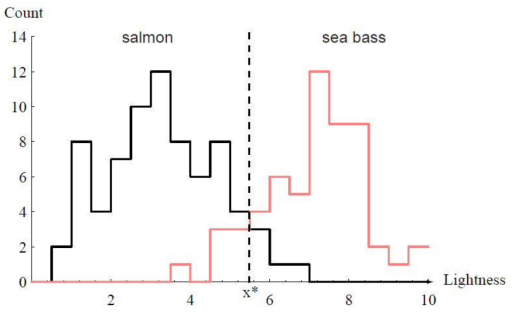
1.4.1 Example: Automatic Fish-Packing Plant



- Pre-processing: apply a segmentation operation in order to isolate fishes from one another and from the background
- Feature extraction: measure some features or properties from the image which will help in discriminating the two species of fish considered (e.g., fish length and width).
- Classification: evaluate the evidence presented and make a final decision as to the species.



(a) Classification Based on Length Feature



(b) Classification Based on Lightness Feature

The two graphs represent density probability obtained by the histogram. It have to be distinguished. No overlap are admitted.
In order to increase further the discrimination capability, one may think to adopt two features instead of just one. For instance, one could consider the width feature

in addition to the lightness one: $fish \rightarrow \mathbf{x} = [x_1, x_2]$, where x_1 represents Lightness and x_2 represents Width.
We need to define classification model and cost function. During the training phase, the classifier should estimate the best linear separation according to the adopted cost function. It is important to not tune excessively (overfit) the classifier so that to leave room for a correct (at least satisfactory) classification of novel patterns (samples). The final objective is generalization.

Chapter 2

Statistical Estimation Theory

2.1 Introduction

Statistical estimation is rather important as it many intervene in different parts of a system based on machine learning. The objective of this chapter is to study methods and algorithms for estimating the statistical distribution of a stochastic signal from a set of available observations (samples).

Statistical estimation can be necessary in pre-processing, feature extraction and recognition. We can use statistical modeling and estimation on the observation space to have classification, detection or prediction outputs.

In the estimation theory, stochastic signals can be subdivided into three categories:

- Noisy deterministic signals: The information source is completely known. The noise interference intervenes during the transmission and/or the acquisition phases (e.g., transmission of signals in telecommunication systems based on PAM).
- Noisy parametric signals: The information source is only partially known. The observations allow estimating the random parameters controlling the behavior of the signal (e.g., target speed velocity estimation in sonar systems).
- Noisy random signals: The signal is completely unknown. In this case, the estimation should rely completely on the available observations (e.g., buried object detection with ground penetrating radar).

Second and third categories are the most common in real applications.

2.2 PDF Estimation

Let $\mathbf{x} = (x_1, ..., x_n)$ be a vector of n features with unknown probability density function (pdf) $p(\mathbf{x})$. Let $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ be a finite set of N independent and identically distributed (iid) samples drawn from the considered pdf. Let us term these samples as training samples. The objective is to determine an estimate $\hat{p}(x)$ on the basis of the available samples $\mathbf{X}$, which is as close as possible to the true proof $p(\mathbf{x})$.

2.2.1 Parametric Estimation

Let us assume that the model of $p(\mathbf{x})$ is characterized by r parameters which define a parameter vector $\theta = (\theta_1, ..., \theta_r)$. The dependency of the model on θ is underlined by the following notation $p(\mathbf{x}|\theta)$.
Since $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ is a vector of iid random variables, we can define a related likelihood function as follows: $p(\mathbf{X}|\theta) = \prod_{k=1}^{N} p(x_k|\theta)$
This function defines the likelihood of θ with respect to the considered set of samples ($\mathbf{X}$). In other words, it provides a useful measure of compatibility/agreement between θ and $\mathbf{X}$.

2.2.2 Estimation Procedures

Main procedures are:

- Maximum Likelihood Estimation: θ is viewed as a vector of quantities whose values are fixed but unknown. The best estimate of their value is defined to be the one that maximizes the probability of obtaining the samples actually observed.
- Bayesian Estimation: The parameters are viewed as random variables having some known prior distribution. Observation of the samples converts this to a posterior density.

2.2.3 Estimation Goodness

The estimate of the vector of parameters depends on the observation vector $\mathbf{X} : \hat{\theta} = \hat{\theta}(\mathbf{X})$. Therefore, the estimate is a random vector.
Let us define the estimation error ϵ as: $\boldsymbol{\varepsilon} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta} = \boldsymbol{\varepsilon}(\mathbf{X}, \boldsymbol{\theta}) = [\hat{\theta}_i - \theta_i; \quad i = 1, 2, ..., r]$
In order to get an ideal estimate for each parameter $\theta_i (i = 1, 2, ..., r)$, it is necessary that each corresponding estimation error ϵ_i is unbiased and has no variance. The ias of an estimation error is its mean value. An estimate is said unbiased if $E\{\epsilon\} = 0 \implies E\{\hat{\theta}\} = \theta$
Its variance is defined as: $var\{\varepsilon_i\} = E\{(\hat{\theta}_i - \theta_i)^2\} \quad (i = 1, 2, ..., r)$
In order to asses the goodness of the variance of our estimate, it is necessary to refer it to the so-called Cramér-Rao bound.

Cramér-Rao Bound

It expresses a lower bound on the variance of an unbiased statistical estimator, based on the Fisher information.
$var(\epsilon\_i) \geq [I^{-1}(\theta)]_{ii}, \quad i = 1, 2, ..., r$ where $I(\theta) = E\left\{ [\nabla_\theta \ln p(\mathbf{X}|\theta)] \, [\nabla_\theta \ln p(\mathbf{X}|\theta)]^T \right\}$
is the Fisher information matix which is defined as: $[I(\theta)]_{ij} = E\left\{ \frac{\partial \ln[p(\mathbf{X}|\theta)]}{\partial \theta_i} \cdot \frac{\partial \ln[p(\mathbf{X}|\theta)]}{\partial \theta_j} \right\}$
Note: When the bound is reached, the estimator is said *efficient*.

2.2.4 Asymptotic Properties

Often, the estimates used in real problems are obtained by biased and inefficient estimators. In order to judge better the goodness of the considered estimator, one analyzes its behavior for large sets of observations. In other words, an estimator is said to be good if it has good asymptotic properties.
An estimate is said to be **asymptotically unbiased** if:
$\lim_{N\to+\infty} E\{\epsilon\} = 0 \implies \lim_{N\to+\infty} E[\hat{\theta}] = \theta$
It is **asymptotically efficient** if: $\lim_{N\to+\infty} \frac{var\{\epsilon_i\}}{[I^{-1}(\theta)]_{ii}} = 1, \quad i = 1, 2, ..., r.$
An estimate is said to be **consistent** if it converges to the true value when $N \to +\infty$:
$\lim_{N\to+\infty} P(||\epsilon|| < \delta) = 1, \quad \forall\delta > 0.$
The necessary condition for consistency is that the estimate is asymptotically unbiased and with variance converging to zero when $N \to +\infty$.

2.3 Maximum Likelihood Estimation

Definition. *The maximum likelihood (ML) estimate of θ is:*

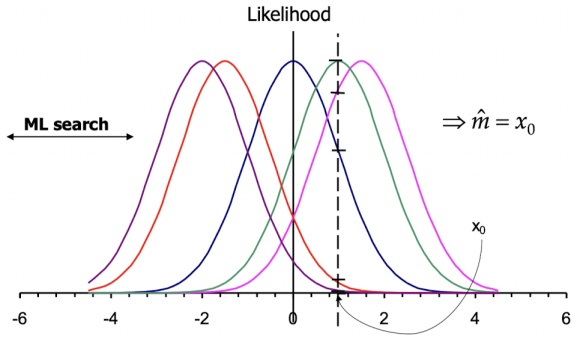
$$\hat{\theta} = \arg\max_{\theta} p(X \mid \theta)$$

By varying θ,one obtains different pdfs, each yielding a likelihood value. Intuitively, the ML estimate corresponds to the value of θ that in some sense best agrees with the actually observed training samples.

For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the value itself. Since the logarithm is monotonically increasing, the θ that maximizes the **log-likelihood** also maximizes the likelihood:

$$\hat{\theta} = \arg\max_{\theta} ln(p(X \mid \theta))$$

`Example:`
ML estimation of the mean of a mono-dimensional Gaussian pdf ($\hat{m}$) with known variance basing on a single observation x_0.



The one which maximize the likelihood is the green one.

Properties

In certain hypotheses regarding $p(\mathbf{x}|\theta)$, it can be shown that, if it exists an efficient estimate and if the ML estimate is unbiased, then the efficient estimate is that provided by ML estimation.

Even if it does not exist an efficient estimate, the ML estimate exhibits good asymptotic properties since it is:

- Asymptotically unbiased
- Asymptotically efficient
- Consistent

Such properties are behind the common use of ML estimation in numerous real problems.

2.3.1 Statistical Model Selection

In the selection of a statistical model for $p(x)$, three main aspects should be considered:

1. The intrinsic statistical nature of the physical phenomenon under analysis;
2. The noise introduced during the (passive/active) signal transmission and acquisition phases by the propagation medium and the sensor, respectively;
3. The pre-processing and feature extraction steps adopted before feeding the recognition system.

Among the popular statistical models, one can find: Gaussian model, Generalized Gaussian model, Gamma model, Rayleigh model, Chi-square model, Log-Normal model.
`Slide 71-73 PDF Examples`

Gaussian Model

The Gaussian model is a parametric model widely used in numerous applications. The motivation behind this success can be found in the central limit theorem which states:
"if the sum of many iid variables has a finite variance, then it will be approximately normally distributed."
Since many real processes take origin from many independent causes and yield distributions with finite variance, this explains the ubiquity of the Gaussian pdf.
The multivariate Gaussian pdf takes the following analytical expression:

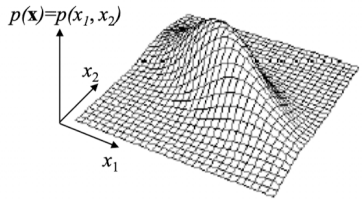
$$p(\mathbf{x}|\theta) = p(\mathbf{x}|\mathbf{m},\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^t \Sigma^{-1}(\mathbf{x}-\mathbf{m})\right]$$

Covariance matrix

Mean vector

N(m, Σ)

Example of a bivariate Gaussian pdf.



$N(m, \sum)$ indicates the Mahalanobis distance: distance between a point and a cloud of points.
The parameters defining completely the multivariate Gaussian pdf are the mean vector and the covariance matrix, which are defined as:

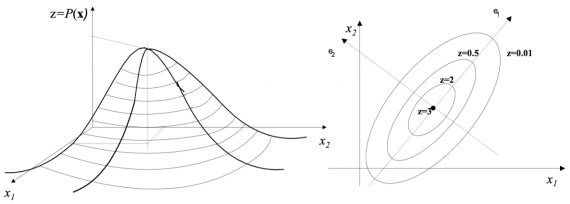
$$\mathbf{m} = E\{\mathbf{x}\}$$
$$\Sigma = Cov(\mathbf{x}) = E\{(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^\top\} = E\{\mathbf{x}\mathbf{x}^\top\} - \mathbf{m}\mathbf{m}^\top$$

The covariance matrix encodes the shape of the distribution.
Mean vector is the center.
Properties of the covariance matrix:

- Σ is symmetric: $\Sigma = \Sigma^\top$.
- Σ is positive semidefinite.
- For independent features: $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix} \quad \Rightarrow \quad p(\mathbf{x}) = p(x_1)p(x_2)\cdots p(x_n),$
  where σ=variance, defined as $\sigma = E\{(x_1 - \bar{x}_1)^2\}$.

Shape of a 2D Gaussian PDF

$p(x)$ can be seen as a bell of unitary volume. The horizontal slices of the bell corresponding to isolevels are ellipses whose axes are directed by the eigenvectors Σ. The eigenvector corresponding to the largest eigenvalue defines the main axis of the ellipse.



Shape of a n-D Gaussian PDF

The previous observations done for the 2-D case can be generalized to the n-D one:

- Let $\lambda_1, ..., \lambda_n$ be the eigenvalues of $\sum$ and $e_1, e_2, \ldots, e_n$ be the corresponding eigenvectors.
- By convention, the eigenvalues and eigenvectors are ordered so that $\lambda_1 \geq \lambda_1 \geq ... \geq \lambda_n$.
- Since $\sum$ is symmetric and positive semidefinite, the eigenvalues will take positive values.
- The eigenvectors will form an orthogonal basis.
- The isolevels of $p(x)$ are hyperellipses in R^n, whose axis directions are governed by the eigenvectors.
- The first eigenvector will define the principal axis while the last one will determine the smallest axis.

NOTE:

$$\text{Covarianza: } Cov_{12} = \frac{1}{n}\sum_{i=1}^{N}(x_{i1} - \overline{x}_1) \cdot (x_{i2} - \overline{x}_2)$$

$$\text{Varianza: } Var(X) = \frac{1}{n}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

Gaussian Model: ML Estimation

It can be shown that the ML estimation of the mean vector and the covariance matrix of a multivariate Gaussian pdf, given a set of N iid training samples $X = \{x_1, ..., x_N\}$ lead to:

$$\text{Mean vector } \hat{m} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k$$

$$\text{Covariance Matrix } \hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}_k - \hat{\mathbf{m}})(\mathbf{x}_k - \hat{\mathbf{m}})^T = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k\mathbf{x}_k^T - \hat{\mathbf{m}}\hat{\mathbf{m}}^T$$

Such estimates are asymptotically unbiased, efficient and consistent.
In addition we have:

$$\text{Estimate of the mean (unbiased) } E\{\hat{m}\} = m$$

$$\text{Estimate of covariance matrix (likeli biased) } E\{\hat{\sum}\} = \frac{N-1}{N} \cdot \sum$$

Steps

We have:

$$P(x) = P(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

With training samples: $X = \{x_1, ..., x_n\}$

STEP 1 Calculate Likelihood function

$$P(\mathcal{X}|\mu, \sigma^2) = \prod_{i=1}^{N} P(x_i|\mu, \sigma^2) \rightarrow Likelihood$$

$$\log P(\mathcal{X}|\mu, \sigma^2) = \sum_{i=1}^{N} \log P(x_i|\mu, \sigma^2) = \sum_{i=1}^{N} \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2}\right] \rightarrow Log{-}likelihood$$

$L(\mu, \sigma)$ depends only on μ and σ

STEP 2 Maximization of $L(\mu, \sigma)$

$$\frac{d^2\mathcal{L}}{d\mu^2} = \sum_{i=1}^{N} \left[-\frac{2(x_i-\mu)\cdot(-1)}{2\sigma^2}\right]$$

$$\text{Find stationary point} \rightarrow \frac{d\mathcal{L}}{d\mu} = 0 \iff \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

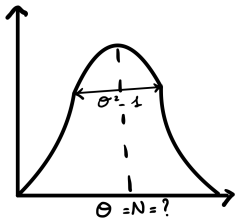
$$\frac{d^2\mathcal{L}}{d\sigma^2} = \sum_{i=1}^{N} \left[-\frac{4}{2\sigma^2} - \frac{(x_i-\mu)^2\cdot(-2)\cdot 1}{2\sigma^4}\right]$$

$$\text{Find stationary point} \rightarrow \frac{d\mathcal{L}}{d\sigma^2} = 0 \iff \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - N)^2$$

2.4 Bayesian Estimation

Bayesian estimation is conceptually different with respect to ML estimation. In ML methods, we view the true parameter vector (θ) we seek to be fixed. In Bayesian learning, we consider θ to be a random variable, and training data allows us to convert a distribution on this variable into a posterior probability density.

NOTES:
$x \sim p(x|\theta)$ is a model that depends on $p(x|\theta)$.



The best value of theta maximize the probability of θ after know training samples X: $\hat{\theta} = argmax\{p(\theta|X)\}$.
Remind that $p(\theta)$ is a prior density, $p(\theta|X)$ is the posterior density.

Bayesian estimation can be applied to any situation in which the unknown density can be parameterized. Its basic assumptions are as follows:

- The form of the density $p(x|\theta)$ is assumed to be known, but the value of the parameter vector θ is not known exactly.
- Our initial knowledge about θ is assumed to be contained in a known a priori density $p(\theta)$.
- The rest of our knowledge about θ is contained in a set X of N samples $x_1, ..., x_N$ drawn independently according to the unknown probability density $p(x)$.

The third assumption can be converted into a posterior density $p(\theta|X)$, which we hope, is sharply peaked about the true value of θ.

The goal is to compute $p(x|X)$, which is as close as possible to $p(x)$. This can be done by:

$$p(x|X) = \int p(x, \theta|X) d\theta$$

By using Bayes theorem:

$$p(x|X) = \int p(x|\theta, X) p(\theta|X) d\theta$$

Since the distribution of x is completely known once we know the value of the parameter vector θ:

$$p(x|X) = \int p(x|\theta) p(\theta|X) d\theta$$

Remind that x is the feature vector, X is the set of training samples.
Since we don't know the exact value of θ, Bayesian estimation directs us to average $p(x|\theta)$ over all possible values of θ. The available observations X exert their influence on $p(x|X)$ through the posterior probability density $p(\theta|X)$.

Thus, the basic problem in Byesian learning is to compute the posterior density $p(\theta|X)$.
For such purpose, one can make use of the following two relationships:
From Bayes formula

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta},$$

where $p(X|\theta)$ is the Likelihood function, $p(\theta)$ is the prior density and $\int p(X|\theta) \cdot p(\theta) d\theta = p(X)$.
From independence assumption

$$p(X|\theta) = \prod_{k=1}^{N} p(x_k|\theta)$$

Bayesian Estimation: Univariate Gaussian Case

In the following, let us consider a simple example of computation of the posterior density $p(\theta|X)$ and the desired probability density $p(x|X)$ by assuming that:

- $p(x|\theta)$ is a univariate Gaussian distribution;
- In $\theta = [\mu, \sigma^2]$, the only unknown parameter is μ.

Accordingly, $p(x|\mu) \sim N(\mu, \sigma^2$. We shall make the further assumption that $p(\mu \sim N(\mu, \sigma^2$ whose parameters are a priori known. Using formula previously seen:

$$p(\mu|X) = \alpha \prod_{k=1}^{N} p(x_k|\mu) p(\mu)$$

where α is a normalization factor.

Since $p(x|\mu) \sim N(\mu, \sigma^2)$ and $p(\mu) \sim N(\mu_0, \sigma_0^2)$, we can obtain:

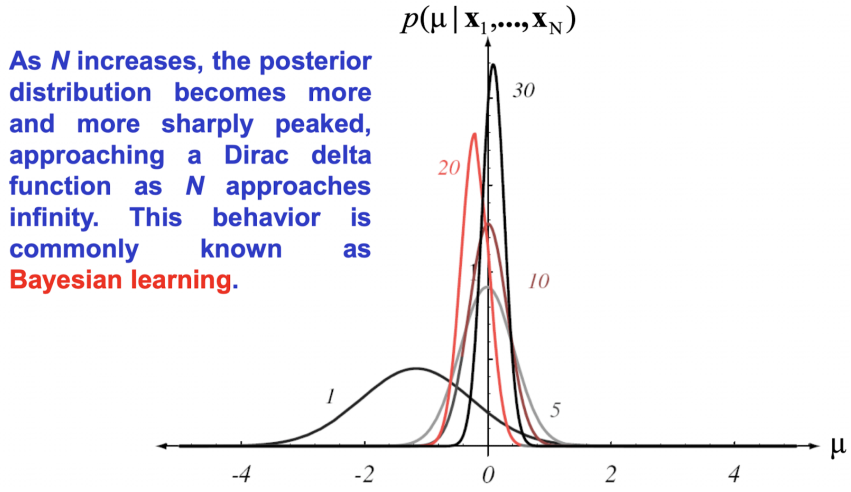
$$p(\mu|X) = \alpha'' \exp\left\{ -\frac{1}{2} \left[\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2} \right) \mu^2 - 2 \left(\frac{1}{\sigma^2} \sum_{k=1}^{N} X_k + \frac{\mu_0}{\sigma_0^2} \right) \mu \right] \right\}$$

where α'' is a constant and $p(\mu|X)$ is again a normal density: $p(\mu|X) \sim N(\mu_N, \sigma_N^2)$.

After some calculations, one can get:

$$\begin{cases} \mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \bar{X} + \left(1 - \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \right) \mu_0, \text{ where } \bar{X} \text{ is Sample mean} \\ \sigma_N^2 = \frac{\sigma^2 \sigma_0^2}{N\sigma_0^2 + \sigma^2} \end{cases}$$

Note: These equations show how the prior information is combined with the empirical information in the samples to obtain the a posteriori density.



Having obtained the a posteriori density for the mean $p(\mu|X)$, all that remains is to obtin the desired density $p(x|X)$:

$$p(x|X) = \int p(x|\mu)p(\mu|X)d\mu = \frac{1}{2\pi\sigma\sigma_N}\cdot\exp\left[-\frac{1}{2}\left(\frac{x-\mu_N}{\sigma^2+\sigma_N^2}\right)^2\right] f(\sigma,\sigma_N), \text{that is a Gaussian}$$

This means that $p(x|X) \sim N(\mu_N, \sigma^2 + \sigma_N^2)$ and thus $\mu = \mu_N$ since $p(x|\mu) \sim N(\mu, \sigma^2)$.

ML vs Bayesian Estimation

The ML approach estimates a point in θ space while the Bayesian approach estimates a distribution $p(x|X)$ from which θ is inferred.

ML and Bayes solution are equivalent in the asymptotic limit of infinite training samples or if the prior $p(\theta)$ is uniform.

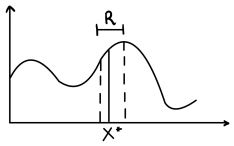
In practice, ML is often preferred over Bayesian estimation because of lower computational complexity and easier interpretability.

NOTE: The outcome of Bayesian is not θ but the distribution itself.

2.5 Nonparametric Estimation

Nonparametric estimation becomes necessary when: there is no prior knowledge about the functional form of the pdf characterizing the observed phenomenon; parametric models do not offer a good approximation of the considered pdf. Nonparametric estimation approaches are thus applied directly on the available observations (training samples).

Basic Concepts



Let x^* be a generic sample and R a predefined region of the feature space such that $x^* \in R$.

Assuming the true pdf $p(x)$ is a continuous function and R is sufficiently small so that $p(x)$ does not vary significantly within it, we can write:

$$P_R = P\{x \in R\} = p(x^* V), \text{ where } V \text{ is the volume of } R.$$

Let K be the number of training samples belonging to R (among a total of N training samples). A consistent estimate of P_R can be achieved through the computation of the relative frequency:

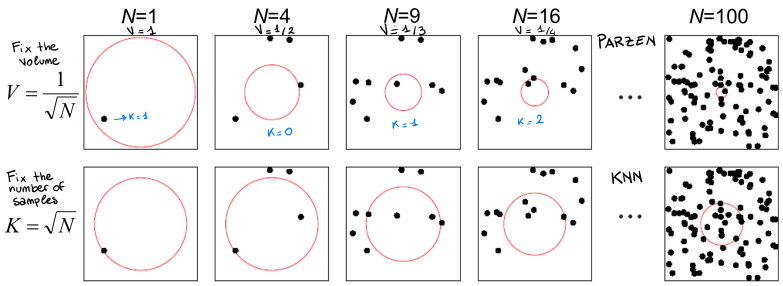
$$\hat{P}_R = \frac{K}{N}, \quad \lim_{N \to \infty} \{|\hat{P}_R - P_R| \leq \delta\} = 1$$

From the estimate of the probability P_R that a sample belongs to R, it can be derived an estimate of the pdf x^*:

$$\hat{p}(x^*) = \frac{\hat{P}_R}{V} = \frac{K}{N \cdot V}$$

Observations:
- R should be enough large to contain a number of training samples that is sufficient for applying the law of large numbers.
- At the same time, it should be enough small to limit the variability of $p(x)$ within it.

Depending on the role taken by each of the two parameters K e V, one may lead to two different nonparametric estimation methods:



KNN: K is fixed and the hypervolume V of R is computed on the basis of the training set to deduce the estimate of the pdf;
Parzen: R (and thus V) is fixed and K is calculated (from the training samples) to determine the pdf estimate.

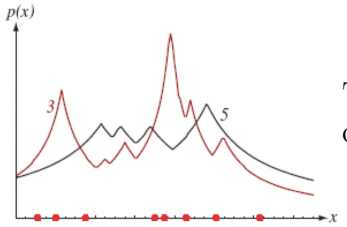
2.5.1 K-NN Estimation

Hypotheses
The number K is set a priori. A shape for the cell of volume centered on x_* is chosen a priori (e.g., hypersphere).
Methodology
The K-NN method consists to expand the cell up to spanning **K** training samples. Let $VK(x^*)$ be the resulting volume. The pdf value at x^* is given by:

$$\hat{p}(x^* = \frac{K}{NV_K(x^*)}$$

It can be shown that, if K is chosen as a function of N, a necessary and sufficient condition to get a consistent estimate in all points, where $p(x)$ is continuous, is given by:

$$\lim_{N\to+\infty} K_N = +\infty, \quad \lim_{N\to+\infty} \frac{K_N}{N} = 0 \rightarrow \text{Example: } K(N) = \sqrt{N}$$



The discontinuities in the slopes in the estimates generally occur away from the positions of the points themselves.

If the choice of K is too large the slope is oversmooth, if K is too small it is undersmooth.

2.5.2 Parzen Windows Estimation

Hypoteses
The Parzen window approach to estimating densities can be introduced by temporarily assuming that R is a n-dimensional hypercube. If h is the length of an edge of that hypercube centered on x^*, then its volume is $V = h^n$.
Metodology
We can obtain an analytic expression for K, the number of samples falling in the hypercube, by defining the following window function:

$$\gamma(x) = \begin{cases} 1, & \text{if } x \text{ belongs to the hypercube} \\ 0, & \text{otherwise} \end{cases}$$

otherwise The training sample x_k belongs to R (with center x^* and edge length h) if $\gamma[(x_k - x^*)/h = 1$. Otherwise, $\gamma[((x_k - x^*)/h] = 0$.
The number of samples in this hypercube is therefore given by:

$$K = \sum_{k=1}^{N} \gamma \left(\frac{X_k - X^*}{h} \right)$$

Which leads to the following estimate:

$$\hat{p}(x^*) = \frac{K}{NV} = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{h^n} \gamma \left(\frac{x_k - x^*}{h} \right)$$

Such estimate is thus as a collection of contributions coming from rectangular functions, each associated with a single training sample.

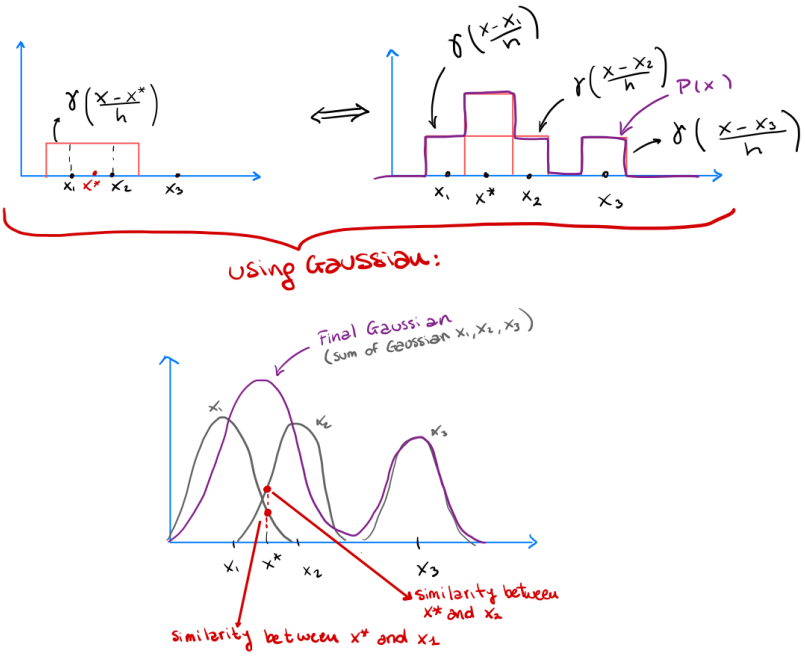
It is also equivalent to a simple count of the number of training samples falling in the predefined hypercube.

This equation suggests a more general approach to estimating density functions.

Rather than limiting ourselves to the hypercube window function, suppose we allow a more general class of window functions.

In such a case, our estimate for $p(x)$ becomes an average of functions of x and the samples x_i:

$$\hat{p}(x) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{V(h)}\gamma\left(\frac{x - x_k}{h}\right)$$



In essence, the window function is being used for interpolation - each sample contributing to the estimate in accordance with its distance from x.

The function $\gamma(\cdot)$ is called Parzen window or kernel and the h parameter is the width of the window (kernel).

It is natural to ask that the estimate $\hat{p}(x)$ be a legitimate density function, i.e., that it be nonnegative and integrate to one.

This can be assured by requiring the window function itself be a density function. To be more precise, if we require that

$$\gamma(x) \geq 0 \quad \forall x \in \mathbb{R}^n, \qquad \int_{\mathbb{R}^n} \gamma(x)dx = 1$$

and if we maintain the relation $V = h^n$, then it follows at once that $p(x)$ also satisfies these conditions. Additional conditions to get a good estimate are:

- $\gamma(\cdot)$ takes a maximal value at the origin
- $\gamma(\cdot)$ is continuous
- $\gamma(x) \to 0$ as $x \to +\infty$, extending distance between x and x^* the similarity decrease.

Window Width effect

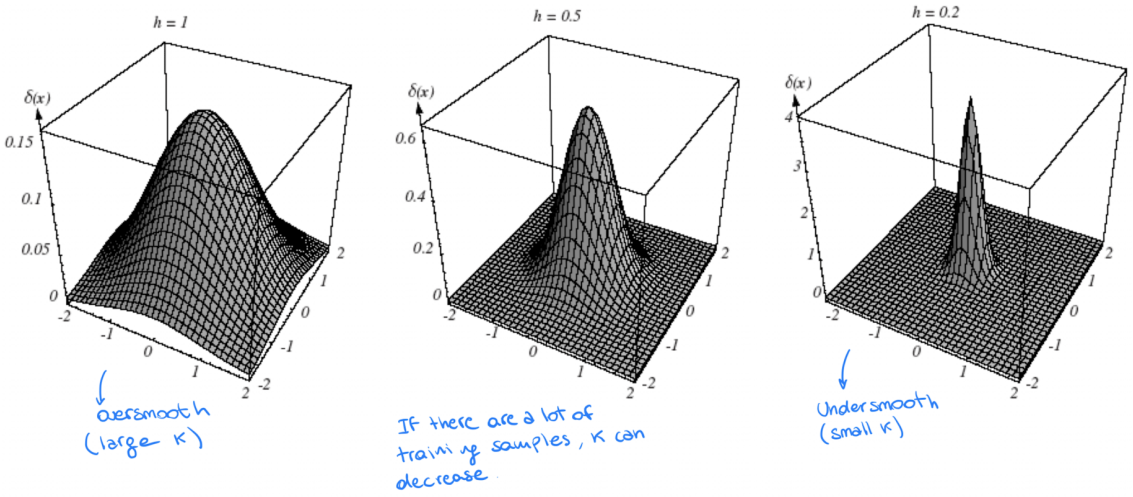
Let us examine the effect that the window width h has on $p(x)$. If we define the function $delta(x)$ by:

$$\delta(x) = \frac{1}{V(h)}\gamma\left(\frac{x}{h}\right)$$

then we can write the estimate for $p(x)$ as the average:

$$\hat{p}(x) = \frac{1}{N}\sum_{k=1}^{N}\delta(x - x_k)$$

`Examples of 2-D circularly symmetric normal Parzen windows for three different values of h`



`Kernel examples`

Parzen window estimation: Properties

Since the samples x_i are iid according to the (unknown) density $p(x)$, we have:

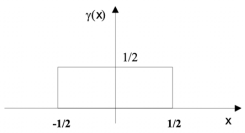
$$E\{\hat{p}(x)\} = p(x) * \frac{1}{h^n}\gamma\left(\frac{x}{h}\right), \quad \textbf{where} * \textbf{represent convolution}$$

The estimate mean is a blurred version of the true $p(x)$.
Parzen estimate is thus biased (bias defined as $\epsilon = \hat{p}(x) - p(x)$). However, it can be shown that it is asymptotically unbiased if the kernel width is chosen opportunely.

$$\lim_{N\to+\infty}\left\{\frac{1}{h_N^n}\gamma\left(\frac{x}{h_N}\right)\right\} = \delta(x) \quad \Rightarrow \quad \lim_{N\to+\infty} E\{\hat{p}(x)\} = p(x)$$

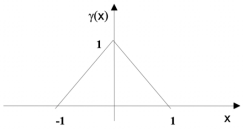
- ***Rectangular kernel***:

$$\gamma(x) = \Pi(x)$$



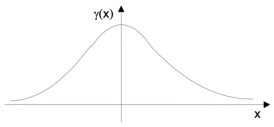
- ***Triangular kernel***:

$$\gamma(x) = \Lambda(x)$$

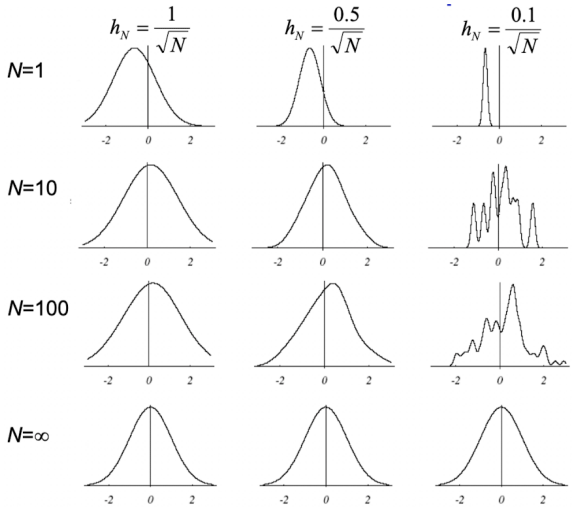


- ***Gaussian kernel***:

$$\gamma(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$



Parzen Windows Estimation: Examples
1-D normal density using different window widths and numbers of samples.



Gaussian Kernel

A commonly used kernel in the Parzen windows estimation is the Gaussian kernel with spherical symmetry (Specht method). In this case, the pdf can be expressed as:

$$\hat{p}(x) = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{(2\pi\sigma^2)^{n/2}} exp\left(-\frac{||x - x_k||^2}{2\sigma^2}\right)$$

The standard deviation σ of the kernel represents thus a smoothing parameter whose setting should be made carefully to avoid both over- and under-fitting.

Since σ is the same along all features, it is important to normalize them before undertaking the pdf estimation process.

Furthermore, one may think to compute σ in an adaptive way, i.e. by associating a σ to each training sample $x_k (k = 1, 2, ..., N)$.

σ_k could be defined as the mean Euclidean distance between x_k and the related L nearest training samples $y_1, ..., y_L$:

$$\sigma_k = \frac{1}{L} \sum_{l=1}^{L} ||x_k - y_1||$$

The Specht method can be implemented by means of Probabilistic **Neural Networks (PNN)**.

2.5.3 Estimation with Incomplete Data

Up to now, we have considered estimation problems where the model depends completely on observable variables. In some real cases, the model depends on unobserved latent variables.

Typically, for problems with incomplete data, the estimation approach that is used is the parametric one.

A solution to this class of estimation problems is the **Expectation- Maximization (EM) algorithm**.

In the following, we will illustrate a typical example of estimation with incomplete data.
Let us assume to have N iid observations $x_i (i = 1, 2, ..., N)$ defined in a n-dimensional space X. Let us assume that these observation are drawn from a distribution $p(x)$ defined as a mixture of M **Gaussian modes**:

$$p(x) = \sum_{i=l}^{M} P_i p(x|m_i, \Sigma_i)$$

where P_i is the prior (prior is given by the heigth of Gaussian) probability (weight), m_i is the mean vector and Σ_i is the covariance matrix of the i-th mixture component.
Let $\Theta = [P_1, ..., P_M;\ m_1, ..., m_M;\ \Sigma_1, ..., \Sigma_M]$ be the vector of parameters to be estimated.

The objective is to estimate the distribution of each Gaussian mode of $p(x)$. Our estimation problem is thus with incomplete data since:

- we have just the observations associated with the whole p(x) and not those of the single modes. We don't know at which Gaussian mode belongs a sample;
- we do not know to which mode is associated each sample (i.e., which are the samples to use for the estimation of each mode?);
- we just know that there is a relationship of the summation type between the different (hidden) modes.

Problem Formulation We have seen that the Maximum Likelihood (ML) estimation problem is given by:

$$\Theta^* = arg\ max\ l(\Theta|X)$$

where

$$l(\Theta|X) = p(X|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta)$$

Depending on the form of $p(X|\Theta)$, the ML estimation problem can be easy or hard. For many problems, it is not possible to find an analytical solution and one must resort to more elaborate techniques. The **Expectation Maximization** (EM) algorithm is one of them.
Let's make the following assumptions. X is observed and generated by some distribution (incomplete dataset). Complete dataset exists: $Z = (X, Y)$ where Y represents missing data The joint density function is given by:

$$p(Z|\Theta) = p(X, Y|\Theta) = p(Y|X, \Theta) \cdot p(X|\Theta)$$

The **complete-data likelihood** function becomes:

$$l(\Theta|Z) = p(Z|\Theta) = p(X, Y|\Theta)$$

While the **incomplete-data likelihood** is:

$$l(\Theta|X) = p(X|\Theta)$$

2.6 Expectation-Maximization Algorithm

The EM algorithm finds the expected value of $log[p(X, Y|\Theta)]$, given X and a current estimate of Θ.
That is, we define the so-called conditional expectation:

$$Q(\Theta, \Theta^{(k)}) = E\{log(p(X, Y|\Theta))|X, \Theta^{(k)}\}$$

$$Q(\Theta, \Theta^{(k)}) = \int log(p(X, y|\Theta))p(y|X, \Theta^{(k)})\, dy$$

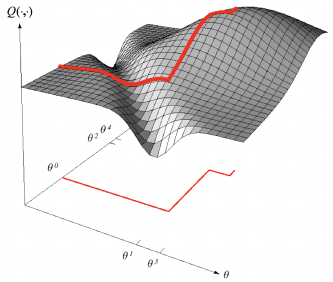
Θ is the value to estimate, Θ^k is a guess of Θ (it could be a random value), $log(p(X, y|\Theta))$ is the log likelihood function.

EM alternates between performing an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated.
EM can thus be formalized into the following two steps:
E-step: $Q(\theta, \theta^{(k)}) = E\{\log(p(X, Y|\theta))|X, \theta^{(k)}\}$

M-step: $\theta^{(k+1)} = \arg\max Q(\theta, \theta^{(k)})$

Convergence of EM Algorithm

It can be shown that the sequence of estimates $\{\Theta^{[k]}\}$ generated by the EM algorithm allows increasing at each iteration the value of the log-likelihood function $l(\Theta)$, i.e.: $l(\Theta^{(i+1)}) \geq l(\Theta^{(i)})$



It is not guaranteed that the EM converges to a global maximum of $L(\Theta)$ (i.e., the ML estimate of Θ). Even though it may suffer from local minima and saddle points, the EM algorithm has proved to be particularly effective in the estimation of mixture components parameters.

Gaussian Mixture

Turning back to the case of a mixture of Gaussian modes, it can be shown that the EM equations for estimating the parameters of each mode are:

$$P_i^{[k+1]} = \frac{1}{N} \sum_{j=1}^{N} \frac{P_j^{[k]} p(x_j|\mathbf{m}_i^{[k]}, \Sigma_i^{[k]})}{p(x_j|\theta^{[k]})}$$

$$\mathbf{m}_i^{[k+1]} = \frac{\sum_{j=1}^{N} \frac{P_i^{[k]} p(x_j|\mathbf{m}_i^{[k]}, \Sigma_i^{[k]})}{p(x_j|\Theta^{[k]})} \cdot x_j}{\sum_{j=1}^{N} \frac{P_i^{[k]} p(x_j|\mathbf{m}_i^{[k]}, \Sigma_i^{[k]})}{p(x_j|\Theta^{[k]})}}$$

$$\Sigma_i^{[k+1]} = \frac{\sum_{j=1}^{N} \frac{P_i^{[k]}\, p(\mathbf{x}_j|\mathbf{m}_i^{[k+1]}, \Sigma_i^{[k]})}{p(\mathbf{x}_j|\theta^{[k]})} (\mathbf{x}_j - \mathbf{m}_i^{[k+1]})(\mathbf{x}_j - \mathbf{m}_i^{[k+1]})^t}{\sum_{j=1}^{N} \frac{P_i^{[k]}\, p(\mathbf{x}_j|\mathbf{m}_i^{[k+1]}, \Sigma_i^{[k]})}{p(\mathbf{x}_j|\theta^{[k]})}}$$

Chapter 3

Feature Reduction

3.1 Introduction

In practical applications, it is not unusual to deal with problems involving tens or hundreds of features.

Intuitively, it may seem that each feature is useful for at least some of the discriminations.

In general, if the performance obtained with a given set of features is inadequate, it is natural to consider adding new features.

Even though increasing the number of features increases the complexity of the classifier, it may be acceptable for an improved performance.

There is a non-zero Bayes error in the 1-D x_1 space or the 2-D x_1, x_2 space. However, the Bayes error vanishes in the 3-D x_1, x_2, x_3 space because of non-overlapping densities.

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** or Hughes effect. 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?

Potential reasons for increase in error include: **wrong assumptions** in model selection and **estimation errors** due to the finite number of training samples for high-dimensional observations (overfitting). Potential solutions include reducing the dimensionality (covariance matrix) and simplifying the estimation.

Objectives of feature reduction are:

- To minimize the implementation cost of the recognition system
- To reduce the computational load of the classifier

- To overcome the Hughes effect

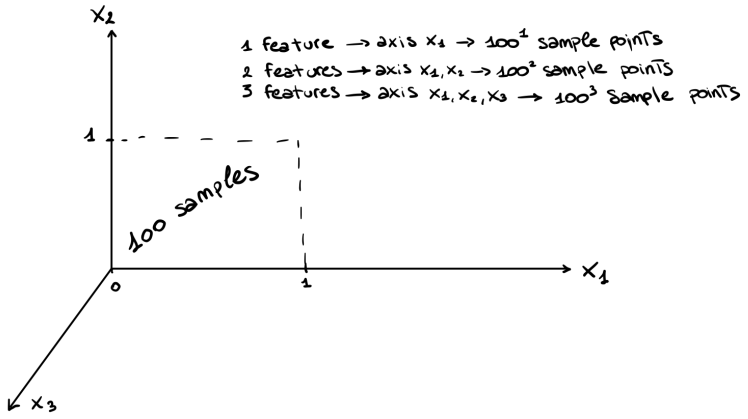
3.2 Hughes Effect

The pdf estimation problem becomes critical when the numbers of training samples and features are unbalanced. The balance depends also on the classifier complexity. Without such balance, the obtained pdf estimate can result unreliable.

The Hughes effect is caused by the exponential increase in volume associated with adding extra dimensions to a given space.

`Example`

- 100 evenly-spaced sample points suffice to sample a unit interval (domain $[0, 1]$) with no more than 0.01 distance between points.
- An equivalent sampling of a 10-dimensional unit hypercube with a lattice with a spacing of 0.01 between adjacent points would require 1020 sample points.
- Thus, in some sense, the 10-dimensional hypercube can be said to be a factor of 1018 "larger" than the unit interval.



Geometrical Analysis

The volume of a hypersphere of radius r in n dimensions is given by:

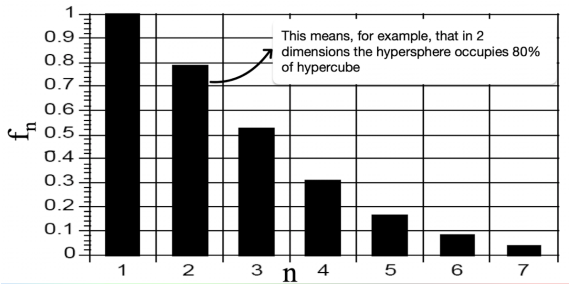
$$V_s(r) = \frac{2r^n}{n}\frac{\pi^{n/2}}{\Gamma(n/2)} \quad \Gamma \text{ is Gamma function}$$

The volume of an hypercube in $[-r, r]^n$ is $V_c(r) = (2r)^n$.
The fraction of the volume of a hypersphere inscribed in a hypercube of the same

dimension then is:

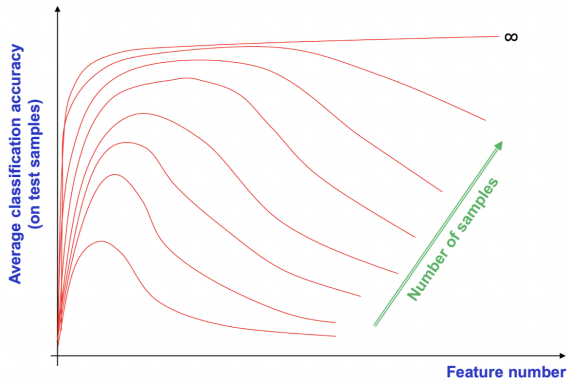
$$f_n = \frac{V_s(r)}{V_c(r)} = \frac{1}{n^{2-1}} \cdot \frac{\pi^{n/2}}{\Gamma(n/2)} \quad \Rightarrow \quad \lim_{n\to\infty} f_n = 0$$

The volume of the hypercube is increasingly concentrated in the corners as n increases. Increasing dimensionality (n-dimensions) of feature space, increase the dis-



tance between training samples.

High dimensional spaces are mostly empty, which implies that multivariate data in R^n is usually in a lower dimensional structure.
Normally distributed data will have a tendency to concentrate in the tails; similarly, uniformly distributed data will be more likely to be collected in the corners, making density estimation more difficult.



3.2.1 Feature Reduction Approaches

Feature reduction aims at: maximizing the number of features while keeping the discrimination capability as higher as possible.
There are two main approaches for feature reduction:
- Feature reduction by selection (**feature selection**)
- Feature reduction by transformation (**feature extraction**)

Feature Selection

Let $F = \{x_1, ..., x_n\}$ be the set of n available features (potential features). In the following, we will denote by f_k (k-th added feature in bottom-up) (and $f_{\underline{k}}$ (k-th

eliminated feature in top-down) the $k-th$ selected and discarded features from F, respectively. The goal is to select a subset F^* composed of $m < n$ features such that:

$$F^* = \underset{F' \subseteq F, \text{card}(F')=m}{\arg\max} \{J(F')\}$$

where $j(\cdot)$ is an opportune function (distance function) that measures the separability between the classes in the space defined by the considered subset of features.
NOTE: At the end, selection problem is an optimization problem.
Feature selection involves two important ingredients:

- A separability measure $J(\cdot)$.
- A search strategy in the solution space. find a combination of indexes of features that maximize $J(\cdot)$.

3.3 Statistical Separability Measures

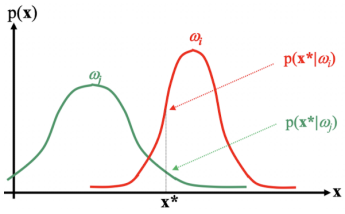
The best separability measure should be compatible with the discrimination criterion adopted by the considered classifier → wrapper methods. A feedback is given by a trivial separability measure: accuracy achieved by the adopted classifier.
An alternative is a measure based on the error probability of the Bayes classifier → embedded methods.
Typical separability measures used in feature selection are:

- Divergence measure
- Bhattacharyya distance
- Jeffries-Matusita distance

3.3.1 Divergence Measure

Let us consider a classification problem with two classes w_i i and w_j characterized by the prior probabilities $P(w_i)$ and $P(w_j)$ and the class-conditional densities $p(x|w_i)$ and $p(x|w_j)$, respectively.
Underlying Idea: measure the separability between the two classes in a considered feature subspace by computing the overlap degree between the two related densities.
In order to define the divergence measure, let us first introduce the so-called likelihood ratio:

$$L_{ij}(x) = \frac{p(x|w_i)}{p(x|w_j)}$$



The divergence measure between the distributions of the two classes is defined as follows:

$$D_{ij}(F') = E\{L_{ij}(x)\} + E\{L_{ij}(\omega)\}$$

where:

$$L_{ij}(x) = \ln[L_{ij}(x)] = \ln[p(x|\omega_i)] - \ln[p(x|\omega_j)]$$

It can be easily verified that:

$$D_{ij}(F') = \int_X [p(x|\omega_i) - p(x|\omega_j)] \ln \left[\frac{p(x|\omega_i)}{p(x|\omega_j)} \right] dx$$

If the class distributions are Gaussian, it can be shown that:

$$D_{ij}(F') = \frac{1}{2}\text{Tr}\left\{(\Sigma_i - \Sigma_j)(\Sigma_i^{-1} - \Sigma_j^{-1})\right\} + \frac{1}{2}\text{Tr}\left\{\Sigma_i^{-1}\Sigma_j^{-1}(\mathbf{m}_i - \mathbf{m}_j)(\mathbf{m}_i - \mathbf{m}_j)^T\right\}$$

where Σ_i, Σ_j and $\mathbf{m}_i$, $\mathbf{m}_j$ are the covariance matrices and the mean vectors of the classes ω_i and ω_j, respectively. Tr$\{\}$ is the matrix trace operator.

Divergence Measure: Properties

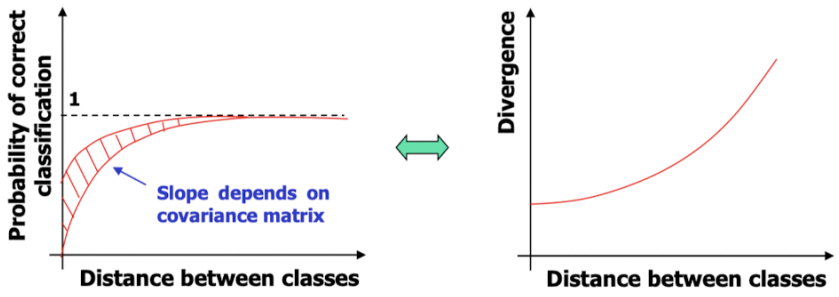
- $\omega_i = \omega_j \Rightarrow D_{ij} = 0$
- $\omega_i \neq \omega_j \Rightarrow D_{ij} > 0$
- $D_{ij} = D_{ji}$ Symmetric
- $D_{ij}(f_1, \ldots, f_k) \leq D_{ij}(f_1, \ldots, f_k, f_{k+1})$ Distance monotonically increases with increasing the number of features

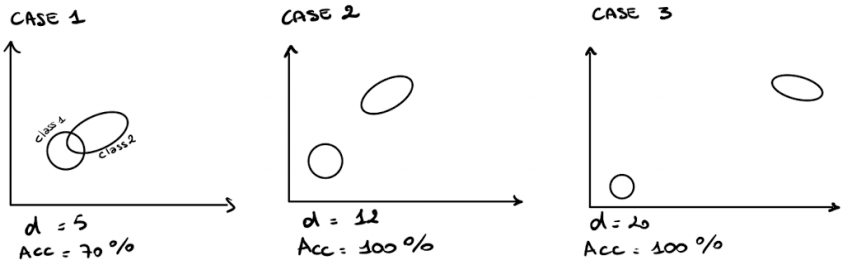
If the features are independent:

$$D_{ij}(f_1, \ldots, f_k) = \sum_{q=1}^{k} D_{ij}(f_q)$$

The larger the divergence, the better the separability between classes.

A drawback of the divergence measure is its non-saturating behavior as the distance between classes increases:





Divergence Measure: Multiclass Case

What has been seen up to now holds for binary classification problems (measure between only 2 classes). For multiclass problems, a multiclass divergence measure could be deduced by:

- Averaging the "binary" divergence measures corresponding to all couples of classes:
$$D_{\text{ave}}(F') = \sum_{i=1}^{C} \sum_{j>i}^{C} P(\omega_i) \cdot P(\omega_j) \cdot D_{ij}(F')$$
- Adopting the worst case reasoning, i.e., by using the lowest binary divergence measure:
$$D_{\min}(F') = \min_{1 \leq i < C;\, i < j} \{D_{ij}(F')\}$$

3.3.2 Bhattacharyya Distance

Objective: definition of a distance measure that depends analytically on the probability of error of the Bayes classifier. The Bhattacharyya distance is defined on the basis of an upper bound of such error probability. In order to define it, let us introduce the Chernoff bound.
The probability of error P_e of the Bayes classifier is given by:

$$P_e = \int_x \{min[P(\omega_i)p(x|\omega_i), P(\omega_j)p(x|\omega_j)]\}dx$$

The direct computation of such quantity is not trivial. To overcome this issue, approximations are necessary.
Using the following relationship:

$$\min[a, b] \leq a^s b^{1-s}, \quad \text{where } 0 \leq s \leq 1$$

we can deduce an upper bound ϵ_u for the Bayes error, called Chernoff bound:

$$\epsilon_u = P(\omega_i)^s P(\omega_j)^{1-s} \int_x p(x|\omega_i)^s \cdot p(x|\omega_j)^{1-s} dx = P(\omega_i)^s P(\omega_j)^{1-s} exp[-\mu_{ij}(s)]$$

The smallest value of ϵ_u, the better the separability between w_i and w_j. The quantity $\mu_{ij}(s)$ is called the **Chernoff distance**.
In case of **Gaussian class distribution**, it can be shown that:

$$\mu_{ij}(s) = \frac{s(1-s)}{2}(m_i - m_j)^t (s\Sigma_i + (1-s)\Sigma_j)^{-1}(m_i - m_j) + \frac{1}{2} ln \left\{ \frac{|s\Sigma_i + (1-s)\Sigma_j|}{|\Sigma_i|^s |\Sigma_j|^{1-s}} \right\}$$

- The Chernoff distance raises the problem of the estimation of the best value of s.
- This can be done empirically so that to maximize $\mu_{ij}(s)$.
- An alternative is to fix arbitrarily the value of s. A particular case is that corresponding to $s = 1/2$. The resulting bound is called **Bhattacharyya bound**:

$$\epsilon_u = \sqrt{P(\omega_i)P(\omega_j)} \int_x \sqrt{p(x|\omega_i) \cdot p(x|\omega_j)} dx = \sqrt{P(\omega_i)P(\omega_j)} exp[-\mu_{ij}(1/2)]$$

In a similar way, the Bhattacharyya distance can be deduced from the Chernoff distance by setting $s = 1/2$. For Gaussian classes, the expression of the Bhattacharyya distance is:

$$B_{ij} = \mu_{ij}(1/2) = \frac{1}{8}(m_i - m_j)^t \left\{ \frac{\Sigma_i + \Sigma_j}{2} \right\}^{-1} (m_i - m_j) + \frac{1}{2} ln \left\{ \frac{|\frac{\Sigma_i + \Sigma_j}{2}|}{|\Sigma_i|^{1/2} |\Sigma_j|^{1/2}} \right\}$$

Properties

- $\omega_i = \omega_j \Rightarrow B_{ij} = 0$
- $\omega_i \neq \omega_j \Rightarrow B_{ij} > 0$
- $B_{ij} = B_{ji}$
- $B_{ij}(f_1, \ldots, f_k) \leq B_{ij}(f_1, \ldots, f_k, f_{k+1})$
- In case of independent features: $B_{ij}(f_1, \ldots, f_k) = \sum_{q=1}^{k} B_{ij}(f_q)$
- No saturating behavior
- Multiclass expression obtainable as done for the divergence distance

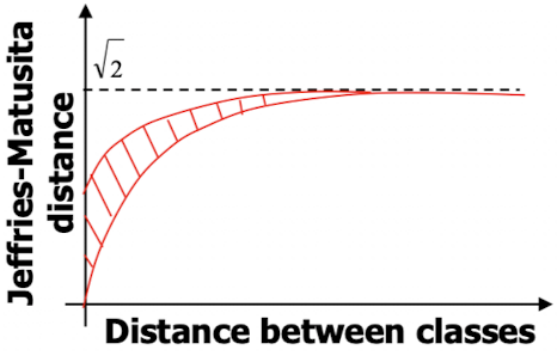
3.3.3 Jeffries-Matusita Distance

Objective: definition of a distance measure with saturating behavior.
The Jeffries-Matusita distance is an average distance between two density functions:

$$JM_{ij} = \left\{ \int_x \left[\sqrt{p(x|\omega_i)} - \sqrt{p(x|\omega_j)} \right]^2 dx \right\}^{1/2}$$

It can be shown that the Jeffries-Matusita distance can be expressed as a functionof the Bhattacharyya distance:

$$JM_{ij} = \sqrt{2(1 - exp(-B_{ij}))}$$



3.4 Sequential Forward/Backward Strategy

Once the separability criterion is adopted, it is necessary to resort to a search strategy in order to identify the best subset of features that optimizes the criterion.

The most intuitive strategy is that based on an exhaustive search. It has the advantage to find the optimal solution but often requires a prohibitive computational cost. Indeed, for a set of n features, the number of subsets of m features ($m < n$), which should be explored, is given by the following binomial coefficient:

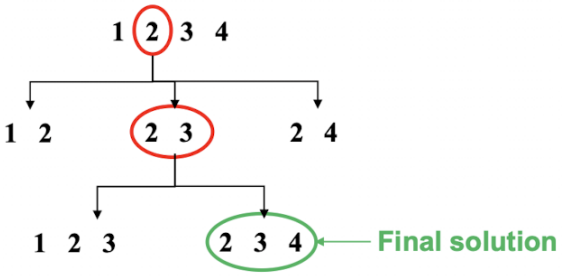
$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

- **Suboptimal strategies:** for saving computational time, they explore just partially the solution space and thus lead to suboptimal solutions.
- **Optimal strategies:** they guarantee a convergence to the optimal solution but are more computationally demanding.

3.4.1 Sequential Forward selection

SFS is an iterative "bottom-up" search strategy, which consists first to find the feature that optimizes singularly the adopted class separability measure. This feature is the first one included in the desired subset of features. At each iteration, the feature which together with those selected in the previous iterations optimizes the adopted criterion, is also selected and added in the subset. This process is iterated up to select the desired number m of features. SFS is suboptimal because of the nesting effect, i.e., a feature previously selected can no more be removed from the subset.

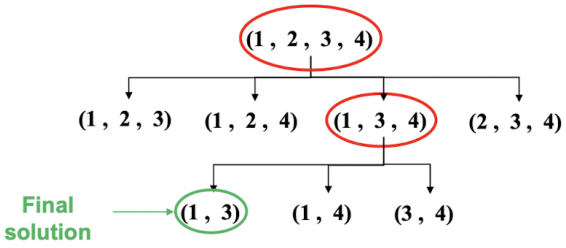
`Example:`
$m = 3$; $n = 4$



3.4.2 Sequential Backward Selection

SBS is said to be a "top-down" search strategy since it works in the opposite direction with respect to that of SFS. At the beginning, all available features are included in the subset. At each iteration, the feature that involves the lowest decrease of the adopted class separability criterion is removed from the subset. This feature removal process continues up to get a subset with cardinality equal to m.

`Example:`
$m = 2$; $n = 4$



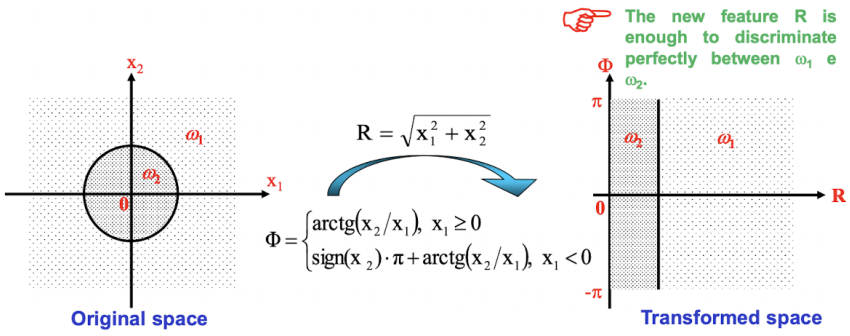
Observations

It's very fast, even with $n >>$.
Feature insertion/removal process is irreversible.

3.5 Feature Extraction

An alternative approach for coping with the feature reduction problem is to combine features. It is often termed as feature extraction.

The potential advantage of feature extraction is to lose less information than feature selection does for a fixed number of desired features.



Issues in feature extraction:

- Linear versus nonlinear transformations
- Use of class labels or not
- Training objective:
  - minimizing classification error (discriminative training)
  - maximizing class separability (linear discriminant analysis)
  - retaining interesting directions (projection pursuit)
  - minimizing reconstruction error (principal component analysis)
  - making features as independent as possible (independent component analysis)

Linear combinations are particularly attractive because they are simple to compute and are analytically tractable. Linear methods project the high-dimensional data onto a lower dimensional space.
Advantages of these projections include:

- Reduced complexity in estimation and classification.
- Ability to visually examine the multivariate data in two or three dimensions.

Given $x \in R^n$, the goal is to find a linear trasformation Φ such that:

$$y = \Phi^t x \in R^m \quad \text{where } m < n$$

Two classical approaches for finding optimal linear transformations are:

- **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.

3.6 Principal Component Analysis

The Principal Component Analysis (PCA) or Karhunen-Loéve transform is an unsupervised feature extraction method frequently used in pattern recognition and signal/image processing applications. Given $x_1, x_2, \ldots, x_N \in R^n$, the goal is to find a m-dimensional subspace where the reconstruction error of x_i in this subspace is minimized. The criterion function for the reconstruction error can be defined in the least-squares sense as:

$$J_m = \sum_{i=1}^{N} \left\| \sum_{k=1}^{m} y_{ik}\phi_k - x_i \right\|^2$$

where $\phi_1, ..., \phi_m$ are the bases for the subspace (stored as the columns of Φ) and y_i is the projection of x_i onto that subspace.

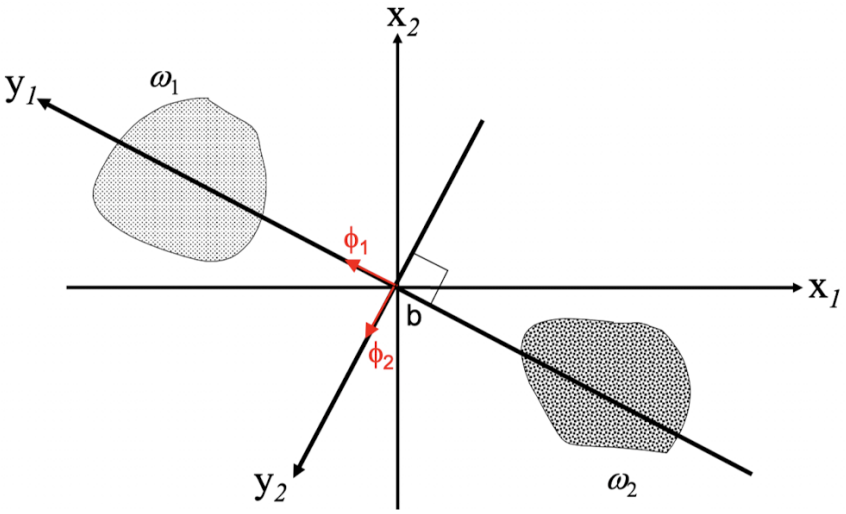
It can be shown that J_m is minimized when $\phi_1, ..., \phi_m$ are the m eigenvectors of the scatter matrix:

$$S = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^t$$

having the largest eigenvalues.
The coefficients $y = (y_1, ..., y_m)^t$ are called the principal components.

When the eigenvectors are sorted in descending order of the corresponding eigenvalues, the greatest variance of the data lies on the first principal component, the second greatest variance on the second component, etc.
Often there will be just a few large eigenvalues, and this implies that the m-dimensional subspace contains the signal and the remaining $n - m$ dimensions generally contain noise.
Note: S is $N - 1$ times the covariance matrix Σ.

Graphical illustration



`Example: SLIDE 175-180`

3.7 Linear Discriminant Analysis

Whereas PCA seeks directions that are efficient for representation, discriminant analysis seeks directions that are efficient for discrimination.

Given $x_1, x_2, \ldots, x_N \in R^n$ divided into two subsets X_1 and X_2 corresponding to the classes ω_1 and ω_2, respectively, the goal is to find a projection onto a line defined as $y = w^t x$ where the points corresponding to X_1 and X_2 are well separated.

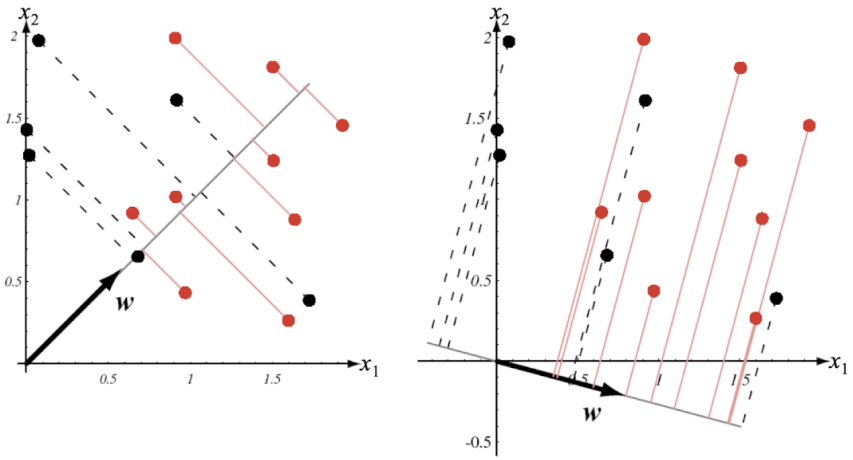


Figure 3.1: Projection of the same set of samples onto two different lines in the directions marked w

A criterion function for best separation could be defined as

$$J(\mathbf{w}) = \frac{\|\widetilde{\mathbf{m}}_1 - \widetilde{\mathbf{m}}_2\|^2}{\widetilde{s}_1^2 + \widetilde{s}_2^2}$$

where $\widetilde{m}_i = \frac{1}{\#(X_i)} \sum_{y \in \omega_i} y$ is the sample mean and $\widetilde{s}_i^2 = \sum_{y \in \omega_i} (y - \widetilde{m}_i)^2$ is the scatter for the projected samples labeled w_i.
This is called the **Fisher's linear discriminant** with the geometric interpretation that the best projection makes the difference between the means as large as possible relative to the variance.

to compute the optimal w, we define:

- The scatter matrices S_i:

$$S_i = \sum_{x \in X_i} (x - m_i)(x - m_i)^t$$

  where

$$m_i = \frac{1}{\#(X_i)} \sum_{x \in X_i} x$$

- The **within-class** scatter matrix S_w:

$$S_w = S_1 + S_2$$

- The **between-class** scatter matrix S_B:

$$S_B = (m_1 - m_2)(m_1 - m_2)^t$$

Then, the criterion function becomes:

$$J(\mathbf{w}) = \frac{\mathbf{w}^t S_B \mathbf{w}}{\mathbf{w}^t S_w \mathbf{w}}$$

and the optimal $\mathbf{w}$ can be computed as

$$\mathbf{w} = S_w^{-1}(m_1 - m_2)$$

Note that S_w is symmetric and positive semidefinite, and it is usually nonsingular if $N > n$. S_B is also symmetric and positive semidefinite, but its rank is at most 1.

LDA: Multiclass Case

Generalization to C classes involves $C - 1$ discriminant functions where the projection is from an n-dimensional space to a $(C - 1)$-dimensional space ($n \geq C$).

- The within-class scatter matrix S_w becomes:

$$S_w = \sum_{i=1}^{C} S_i$$

- The between-class scatter matrix S_B is:

$$S_B = \sum_{i=1}^{C} [\#(X_i)](m_i - m)(m_i - m)^t$$

  where m is the global mean vector:

$$m = \frac{1}{N} \sum_{x \in X} x$$

The criterion function is given by:

$$J(\mathbf{W}) = \frac{\mathbf{W}^t S_B \mathbf{W}}{\mathbf{W}^t S_w \mathbf{W}}$$

where $\mathbf{W}$ is the n-by-$(C-1)$ transformation matrix and $|\cdot|$ represents the determinant.

It can be shown that $J(\mathbf{W})$ is maximized when the columns of $\mathbf{W}$ are the eigenvectors of $S_w^{-1} S_B$ having the largest eigenvalues.

Because S_B is the sum of C matrices of rank one or less, and because only $C-1$ of these are independent, S_B is of rank $C-1$ or less. Thus, no more than $C-1$ of the eigenvalues are nonzero.

`Example: SLIDE 188-191`

Chapter 4

Supervised Classification

4.1 Introduction

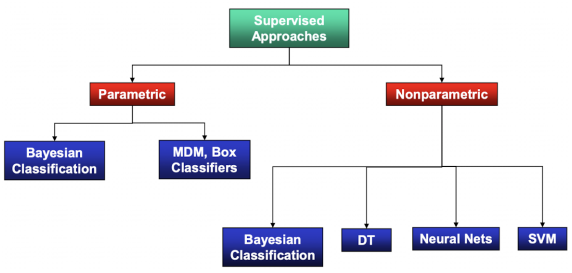
Depending on the availability or not of training samples, one may resort to supervised or unsupervised machine learning techniques. In this chapter, we will deal with the first category of techniques. The design of a supervised classifier depends on:

- the available set of features;
- the available set of training samples;
- the typology of the adopted classification model;
- the cost function to optimize.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector defined in an n-dimensional feature space.
Let Ω be a set of C classes ω_i $(i = 1, 2, \ldots, C)$.
Let 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.



4.2 Parametric Classifier

4.2.1 MDM classifier

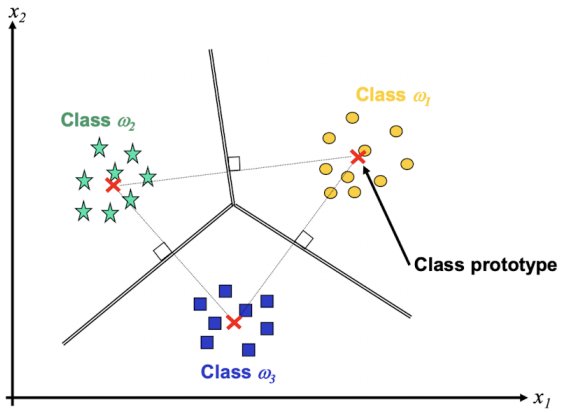
Underlying Hypotheses:

- 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.

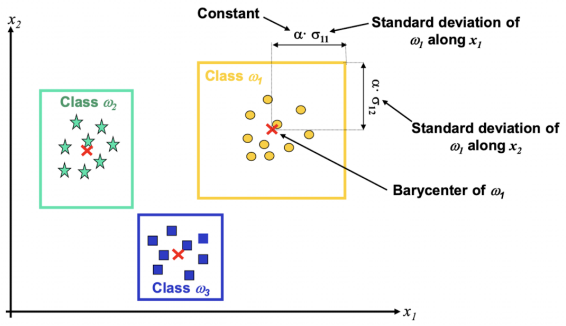


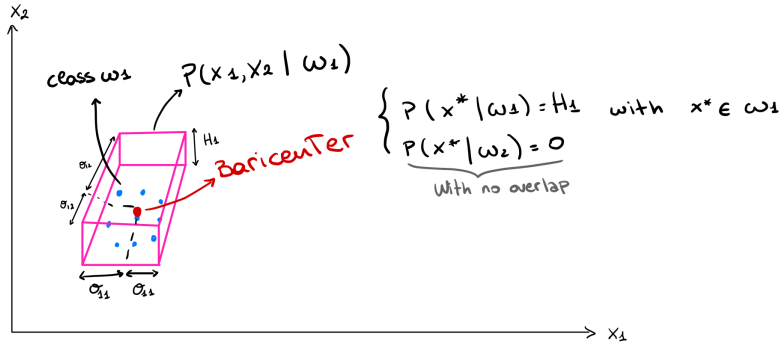
4.2.2 Box classifier

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.

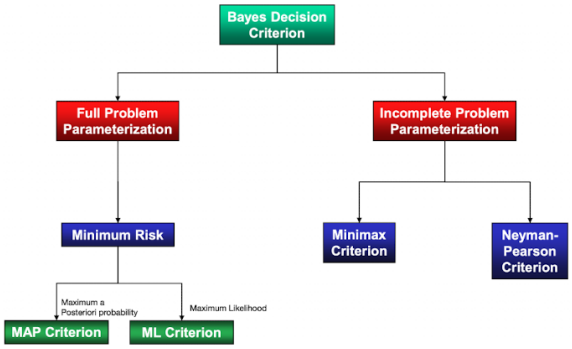
In practice, the resulting boxes can be seen as decision regions. A given unknown pattern is classified according to the box it belongs to.





4.3 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.



4.3.1 MAP Criterion

Hypothesis: a posteriori probabilities of classes $P(w_i|x)$ $(i = 1, 2, ..., C)$ are assumed to be known.

Goal: minimize the average probability of error.

Decision Rule: a pattern x is assigned to the class that maximizes the a posteriori probability $P(w_i|x)$:

$$x \in \omega_j \iff P(\omega_j|x) \geq P(\omega_i|x), \forall i = 1, 2, ..., C$$

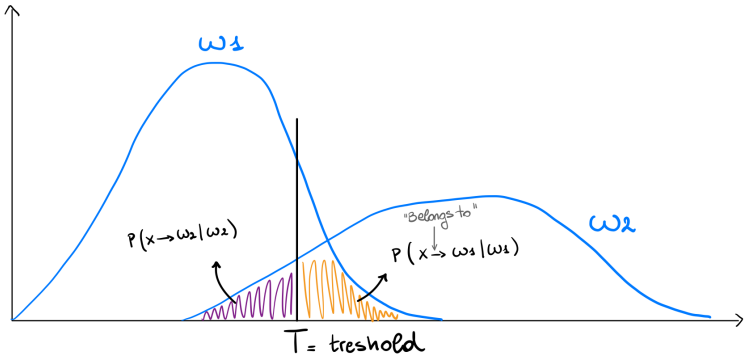
Since the posterior probabilities are often not directly known, it is preferable to rewrite MP decision rule by using Bayes theorem ($posterior = \frac{likelihood \cdot prior}{evidence}$) as follows:

$$x \in \omega_j \iff P(\omega_j|x) \geq P(\omega_i|x), \forall i = 1, 2, ..., C$$

Let R_i be the decision region generated by an arbitrary classifier for the class w_i. The average probability of error is given by:

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

where $\sum_{i=1}^{C} P(err|\omega_i)P(\omega_i)$ is given by the Total Probability theorem.
Posterior probability is correlated with probability of error.



4.3.2 ML Criterion

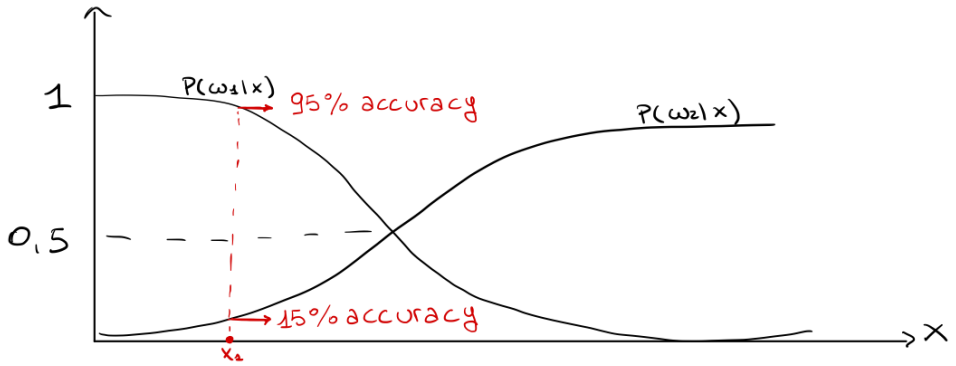
Hypothesis: only the class-conditional pdfs $p(x|w_i)$ $(i = 1, 2, ..., C)$ are assumed to be known.

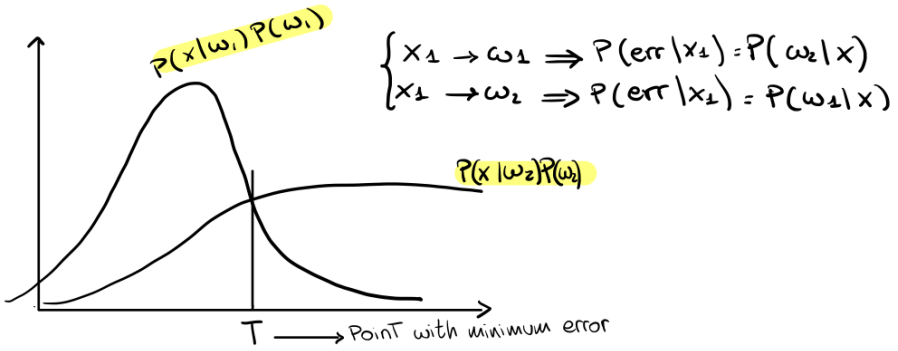
Goal: minimize the average probability of error.

Decision Rule:

$$x \in \omega_j \iff p(x|\omega_j) \geq p(x|\omega_i), \forall i = 1, 2, ..., C$$

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





4.4 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.s The cost of an action depends on the true class to which a considered observation belongs. It is thus

possible to define a $R \cdot C$ cost matrix Λ:

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

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

`Example`
- $\Omega = \{\omega_1 = \text{"fire"}, \omega_2 = \text{"no fire"}\}$
- $A = \{\alpha_1 = \text{"do call fireman"}, \alpha_2 = \text{"do not call firemen"}\}$

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

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).
Conditional Risk:
For each pattern x, let us introduce the conditional risk (expected loss) of taking action α_i:

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

Bayes Risk:
Given x, the action α_j which minimizes the conditional risk is chosen:

$$x\alpha \quad \alpha^* = \alpha_j \iff \quad R(\alpha_j|x) \leq R(\alpha_i|x), \quad \forall i = 1, 2, ..., R$$

The resulting overall risk $R^* = \int_x R(\alpha^*|x)p(x)dx$ is called the Bayes risk and is the best performance that can be achieved.
`Slide 211-213 Example`

Every decision criterion is reduced to $\Lambda(x) = \frac{P(x|\omega_1)}{P(x|\omega_2)} <> \tau$

- If $\tau = 1 \rightarrow$ ML
- If $\tau = \frac{P(\omega_2}{P(\omega_2} \rightarrow$ MAP
- If $\tau = \frac{P(\omega_2}{P(\omega_2} \cdot \frac{(\lambda_{2i} - \lambda_{2(i+1)})}{(\lambda_{1j} - \lambda_{1(j+1)})} \rightarrow$

4.5 Discriminant Functions

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

$$g_i(x) > g_j(x) \quad \forall j \neq i$$

These functions divide the feature space into C decision regions $R_1, ..., R_C$, separated by decision boundaries. For the classifier that minimizes conditional risk:

$$g_i(x) = -R(\alpha_i|x)$$

For the classifier that minimizes error:

$$g_i(x) = P(\omega_i|x)$$

NOTE: $P(\omega_i|x)$ is proportional to $P(\omega_1|x)P(\omega_1)$
NOTE: Results do not change even if we replace every $g_i(x)$ by $f(g_i(x))$ where $f(\cdot)$ is a monotonically increasing function (e.g. logarithm).

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:

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

with $i = 1, ..., C$.
In this case, the discriminant function is written as:

$$g_i(x) = ln\ P(\omega_1|x) + ln\ P(\omega_1)$$

And the discrimination function is:

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

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

The distribution is spherically symmetric.
Discriminant functions are

$$g_i(x) = w_i^t x + w_{i0}$$

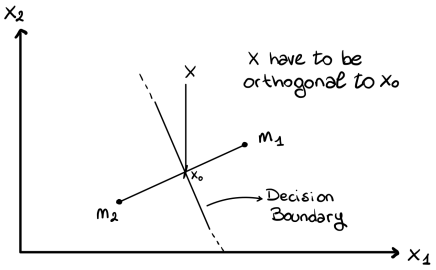
It is a **linear discriminant**. w_{i0} is the bias. Where

$$\begin{cases} w_i = \frac{1}{\sigma^2}m_i \\ w_{i0} = -\frac{1}{2\sigma^2}m_i^t m_i + \ln\ P(\omega_i) \end{cases}$$

Decision boundaries are the hypersurfaces corresponding to $g_i(x) = g_j(x)$.
In this case, they can be written as $w^t(x - x_0) = 0$, where:

$$\begin{cases} \mathbf{w}^t = \mathbf{m}_i - \mathbf{m}_j \\ x_0 = \frac{1}{2}(\mathbf{m}_i + \mathbf{m}_j) - \frac{\sigma^2}{\|\mathbf{m}_i - \mathbf{m}_j\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}(\mathbf{m}_i - \mathbf{m}_j) \end{cases}$$

Hyperplane separating R_i and R_j passes through the point x_0 and is orthogonal to the vector w. The intersection of hyperplane and feature space is the boundary.



$$f_{12} = g_1(x) - g_2(x) \begin{cases} > 0 \rightarrow x \in \omega_1 \\ = 0 \rightarrow \text{Decision Boundary} \\ < 0 \rightarrow x \in \omega_2 \end{cases}$$

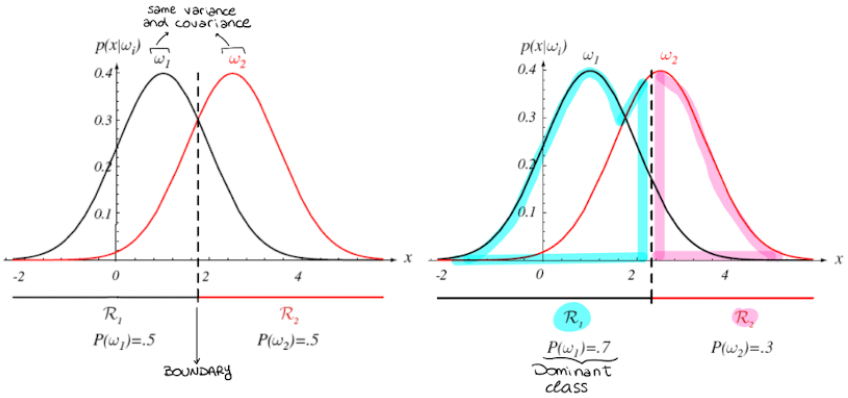


Figure 4.1: Examples with 1-D distributions

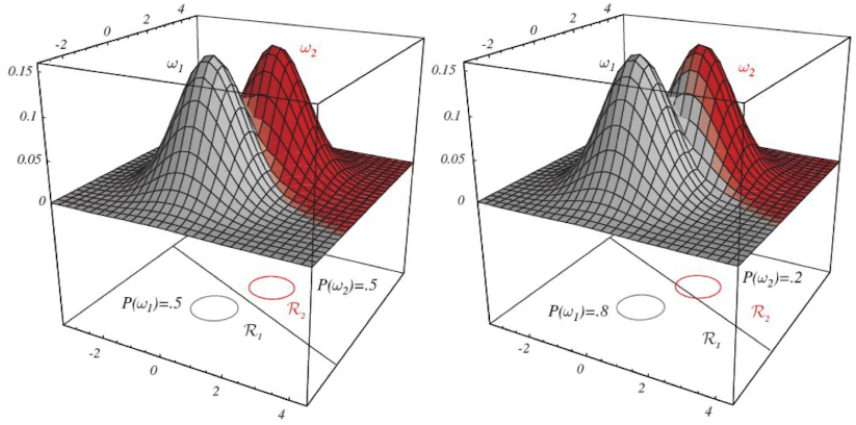


Figure 4.2: Example with 2-D distributions

Case 2: $\Sigma_i = \Sigma$

Isolevels are ellipsis and not spherical.
Discriminant functions become:

$$g_i(x) = w_i^t x + w_{i0}$$

This is a **linear discriminant**. Where:

$$\begin{cases} \mathbf{w}_i = \mathbf{\Sigma}^{-1}\mathbf{m}_i \\ w_{i0} = -\frac{1}{2}\mathbf{m}_i^t\mathbf{\Sigma}^{-1}\mathbf{m}_i + \ln P(\omega_i) \end{cases}$$

where Σ^{-1} indicates that we apply a rotation of the hyperplane passes through x_0 because there's not spherical isolevels.
Decision boundaries are $w^t(x - x_0) = 0$, where

$$\begin{cases} \mathbf{w} = \mathbf{\Sigma}^{-1}(\mathbf{m}_i - \mathbf{m}_j) \\ x_0 = \frac{1}{2}(\mathbf{m}_i + \mathbf{m}_j) - \frac{\ln\left[\frac{P(\omega_i)}{P(\omega_j)}\right](\mathbf{m}_i - \mathbf{m}_j)}{(\mathbf{m}_i - \mathbf{m}_j)^t\mathbf{\Sigma}^{-1}(\mathbf{m}_i - \mathbf{m}_j)} \end{cases}$$

Hyperplane passes through x_0 but is not necessarily orthogonal to the line between the means.

Case 3: $\Sigma_i = arbitrary$

Dicriminant functions are $g_i(\mathbf{x}) = \mathbf{x}^t\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^t\mathbf{x} + w_{i0}$. This is a **quadratic discriminant**. Where

$$\begin{cases} \mathbf{W}_i = -\frac{1}{2}\mathbf{\Sigma}_i^{-1} \\ \mathbf{w}_i = \mathbf{\Sigma}_i^{-1}\mathbf{m}_i \\ w_{i0} = -\frac{1}{2}\mathbf{m}_i^t\mathbf{\Sigma}_i^{-1}\mathbf{m}_i - \frac{1}{2}\ln|\mathbf{\Sigma}_i| + \ln P(\omega_i) \end{cases}$$

Decision boundaries are hyperquadrics (can have different shapes).

If we have to distinguish between multiple classes, we have to compute decision boundaries between all the classes.
`Example slide 232-233`

4.6 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.
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.

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. Each node represent a question and at every node we have to determine the feature and the value (to compare). Links must be mutually distinct and exhaustive. Leaf nodes bear a category labels.
Decision trees bring several interesting properties: interpretability, fast classification and easy incorporation of prior knowledge from human experts.

4.6.1 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:

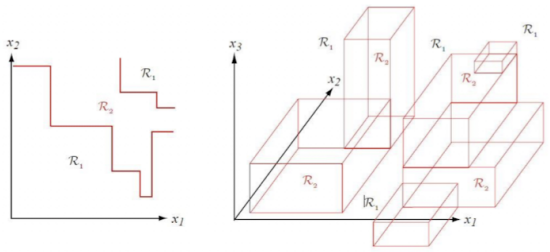
- Number of splits
- Test selection and node impurity
- When to stop splitting
- Pruning
- Assignment of leaf node labels

Number of splits

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.

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.



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

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.

- Entropy impurity: $i(N) = -\sum_j P(\omega_j) log_2(P(\omega_j))$
  $\rightarrow P(\omega_j)$ Fraction of patterns at node N belonging to ω_j
- Variance impurity: $i(N) = P(\omega_1)P(\omega_2)$ with only 2 classes
- Gini impurity: $i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = 1 - \sum_j P^2(\omega_j)$
- Misclassification impurity: $i(N) = 1 - max\ P(\omega_j)$

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:

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

where N_L is the left descendent node and N_R is the right descendent node.
The best test value s is the choice for T that maiximizes $\Delta i(T)$. The optimization is local (done at a single node). The re is no guarantee the successive locally greedy optimal decisions lead to the global optimum.

When to stop 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

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.

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.

Assignment of leaf node labels

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.

`Example slide 249-250`

4.6.2 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.

For a classifier with 2 features, for example, $g(x_1, x_2) = w_1x_1 + w_2x_2 + bias \geq 0$.

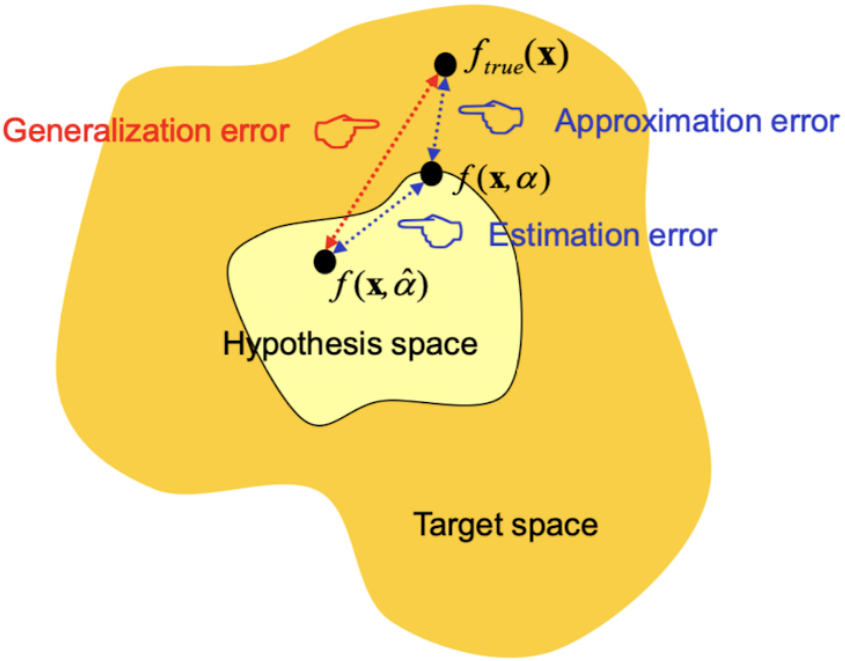
`Example slide 252`

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: $ID3$ and $C4.5$. It is noteworthy that instability issues in de-

cision trees gave birth to an interesting approach called random forest (where each tree represents a classifier), which has been showing very effective.

4.7 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:



Where $f(x, \alpha)$ is the best achievable model and $f(x, \hat{\alpha})$ is the model obtained after the training phase.

Let us consider a machine whose task is to learn the mapping $x_i \rightarrow y_i$, assuming that there exists some unknown probability distribution $P(x, y)$ 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 $x \mapsto f(x, \alpha)$, where the functions $f(x, \alpha)$ themselves are labeled by the adjustable parameters α. The machine is assumed to be deterministic, i.e., for a given input x, and choice of α , it will always give the same output $f(x, \alpha)$. A particular choice of α generates what we will call a trained machine.

The machine is tested on all the combinations of (x_i, w_j) and calculating the error.

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

$$R(\alpha) = \int_{X \times Y} L(y, f(x, \alpha)) p(x, y) dx dy$$

where $L(y, f(x, \alpha))$ is the Loss function.

This is a nice way of writing the true mean error, but unless we have an estimate

of what $p(x, y)$ is, it is not very useful. The quantity $R(\alpha)$ is called the **expected risk**, or just the risk.

4.8 Accuracy Evaluation

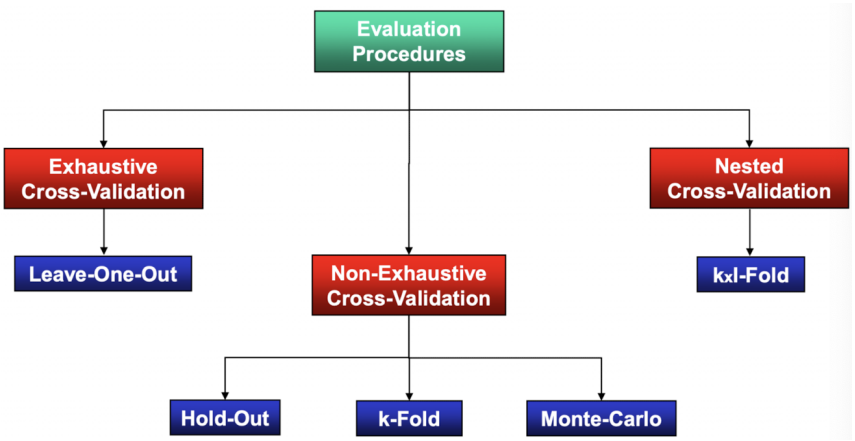
Empirical risk $R_{emp}(\alpha)$ is defined to be the measured average loss (e.g., mean error rate) on the training set (for a fixed, finite number of observations):

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

This equation is an approximation of the generalization error and it 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.



4.8.1 Binary Classifiers

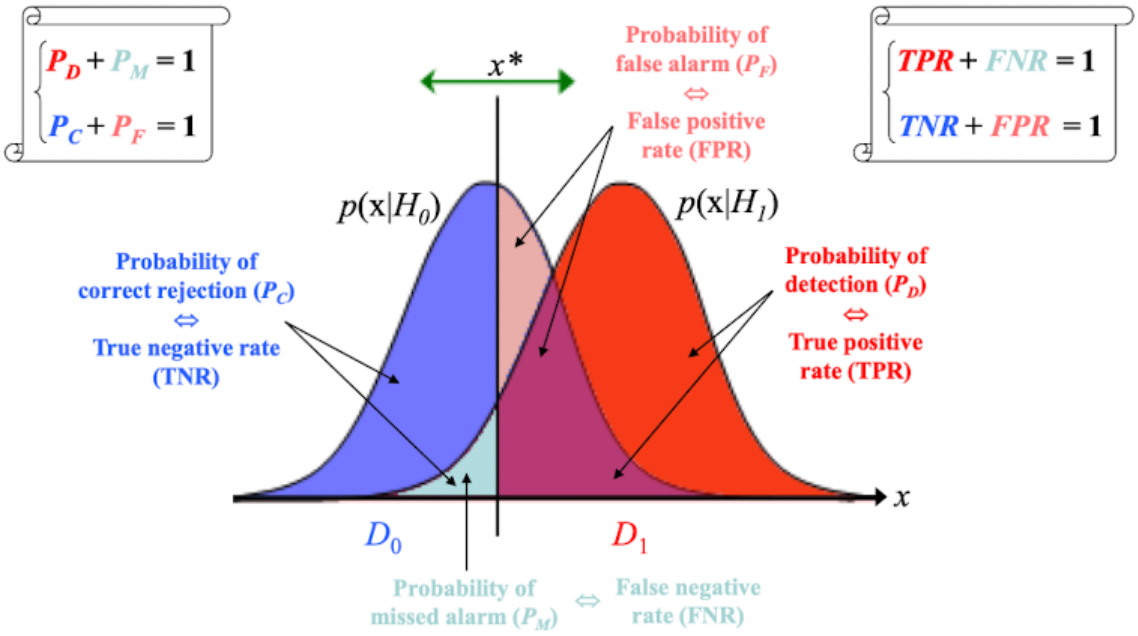


Figure 4.3: Illustration of various probabilities for 1-D normal distributions

Confusion matrix and Common derivation

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

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

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

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

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

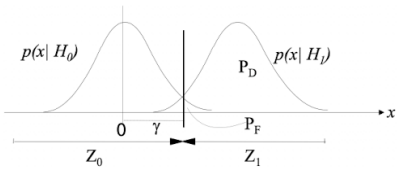
$$F_\beta = \frac{(1+\beta^2)(Precision \cdot Sensivity)}{\beta^2 Precision \cdot Sensivity}$$

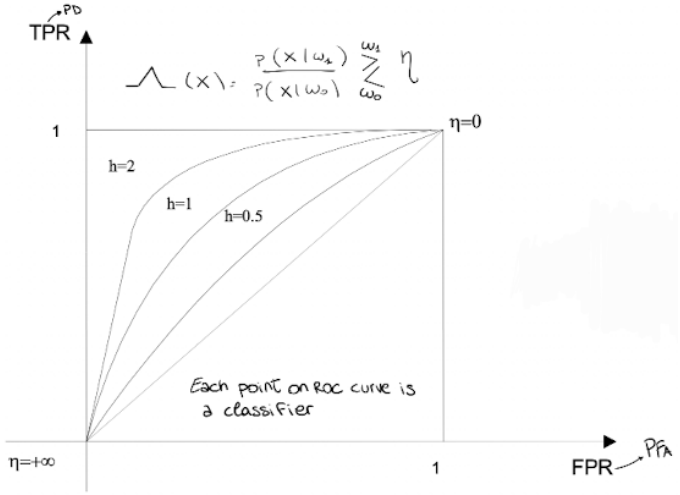
$$F_1 = \frac{2 \cdot Precision \cdot Sensivity)}{Precision + Sensivity}$$

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 $p(x|H_0)$ and $p(x|H_1)$. It depends neither on the cost matrix nor on the priors.

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 parametrized in terms of $h = m/\sigma = 0.5, 1, 2, ...$





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

Confusion Matrix and overall Derivations

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

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

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

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

4.9 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 H_1 called alternative. The objective is thus to know when the differences in H_0 are due to randomness or not.

4.9.1 T-Test

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

Fold	C1	C2	Δ
Fold 1	92.4	92.0	+0.4
Fold 2	91.8	90.6	+1.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Fold i	A11	A21	Δ_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$
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.

$$\text{T-value } T = \frac{\text{signal}}{\text{noise}} = \frac{\text{difference between group means}}{\text{variability of groups}}$$

In our case:

$$\Delta = \frac{1}{k}\sum_{i=1}^{k}(A_{1i} - A_{2i}) = \frac{1}{k}\sum_{i=1}^{k}\Delta_i$$

and

$$Var(\Delta) = \frac{1}{k-1}\sum_{i=1}^{k}(\Delta_i - \overline{\Delta})^2$$

The combination results:

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

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

$$T \leq T_c \begin{cases} \text{Accept } H_0 \rightarrow \\ \text{Reject } H_0 \end{cases}$$

We accept H_0 because the difference of accuracy between classifiers C_1 and C_2 is statistically insignificant.
T_c depends on the desired confidence level α (e.g., 5%) and degree of freedom df ($df = k - 1$).

4.9.2 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

$\Rightarrow$

	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.
The McNemar's test statistic is given by:

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

Under $H_0 : N_{01} = N_{10}$, 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.