NUMERO: 2422A ANNO: 2019

A P P U N T I

STUDENTE: Gentili Cristina

MATERIA: Machine learning and artificial intelligence - Prof. Caputo

Machine learning and Artificial Intelligence

POSSIBLE DEFINITIONS:

AI: theory and algorithms that enable computers to mimic human intelligence

ML: a subset of AI that includes statistical techniques enabling machines to improve at tasks with experience

This course focuses on a specific family of ML algorithms (DISCRIMINATIVE METHODS)

Basic Statistics

▶ PROBABILITY

DEF Sample Space

▶ A Sample Space Ω is the set of all possible outcomes of a (conceptual or physical) random experiment.

[Ω can be finite or infinite]

All the possible outcomes of my experiment is a sample space.
Every ~~measurement~~ measure in my experiment is a random variable.
Because of it's a random variable it has an intrinsic uncertainty
→ always there is a level of uncertainty,
a level of probability

examples:
1) Ω may be the set of possible outcomes of a dice roll (1, 2, 3, 4, 5, 6)
2) pages of a book opened randomly (es: 1-157)
3) Real numbers for temperature, location, time ... and so on

Given the definition of Sample Space, we are now interested in EVENTS

⇒ We will ask the QUESTION:
What is the probability of a particular event?

DEF Event:

▶ Event A is a subset of the sample space Ω

6- additivity: $P\left(\bigcup_{i=1}^{M} A_i\right) = \sum_{i=1}^{M} P(A_i)$

If the events within the sample space are disjoint, the probability of the union of all the possible events is equal to the sum of the probability of the events

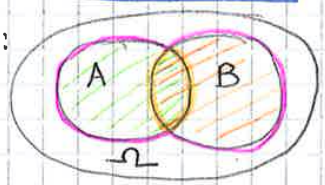
example:
So, in the example of the dice → the probability of the union of all the possible outcomes that I have rolle in the dice: (1,2,3,4,5,6) is equal to the probability that I get 1, plus the probability that I get 2, plus... and so on...

⇒ consequences:

- $P(\emptyset) = 0$
  the probability of the empty event is zero
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
  ↳ the probability of the union between the events A AND B
  OSS ↳ if the 2 events A AND B are DISJOINT, the probability of the intersection between them is zero: $P(A \cap B) = 0$
- $P(A^c) = 1 - P(A)$
  ↳ the probability of the complement of A

Venn Diagram

ex:



$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$P(A \cap B)$ is $\neq 0$ if A AND B (partially) overlap

Random Variables

→ A RANDOM VARIABLE X is a variable whose possible values are numerical outcomes of a random PHENOMENON

DEF ▸ Real valued RANDOM VARIABLE is a function of the outcome of a randomized experiment

$X : \Omega \longrightarrow \mathbb{R}$

- X: random variable
- Ω: sample space
- ℝ: space of real numbers

A RANDOM VARIABLE IS DEFINED AS A FUNCTION that MAPS the OUTCOMES OF AN UNPREDICTABLE PROCESS to a numerical QUANTITIES (Real numbers)

- $P(a < X < b) = P(\omega : a < X(\omega) < b)$
- $P(X = a) = P(\omega : X(\omega) = a)$
  ↓ such that

Examples:

- DISCRETE RANDOM VARIABLES examples (Ω is discrete):
  1) ex 1
     $X(\omega)$ = true if a RANDOMLY DRAWN person(ω) from our class (Ω) is female
  2) ex 2
     $X(\omega)$ = ~~[illegible]~~ is the hometown $X(\omega)$ of a randomly drawn person (ω) from our class (Ω)

Continuous Distribution

DEF: CONTINUOUS PROBABILITY DISTRIBUTION: its cumulative distribution function is absolutely continuous

DEF: CUMULATIVE DISTRIBUTION FUNCTION

1) USA → ANS IN ASE the closed

$$F_X(z) = P(X \leq z)$$

2) HUNGARY

$$F_X(z) = P(X < z)$$

OSS **DEF**: let $F(-\infty) = 0$ AND $F: (-\infty, \infty] \to \mathbb{R}$ IS ABSOLUTELY CONTINUOUS

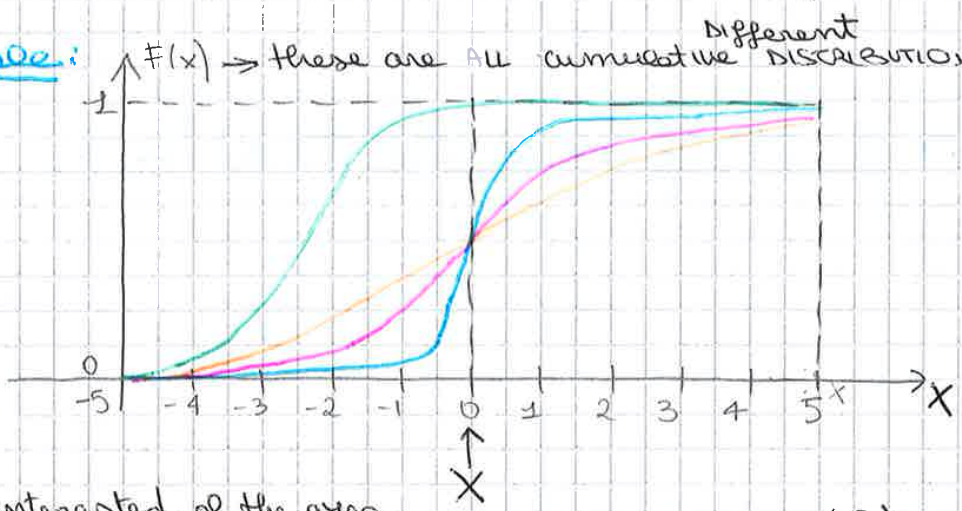
$$\Rightarrow \boxed{F(x) = \int_{-\infty}^{x} f(t)\,dt} \text{ for some function } f$$

cumulative DISTRIBUTION function

DEF: f is called the DENSITY of the DISTRIBUTION

Properties: 1) $\frac{d}{dx} F(x) = f(x)$ 2) $F(x) = \int_{-\infty}^{x} f(t)\,dt$

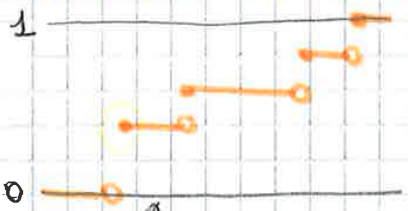
examples: $F(x)$ → these are ALL cumulative DISTRIBUTION FUNCTIONS



I am interested of the area below the FUNCTION until the POINT of INTEREST X (?)

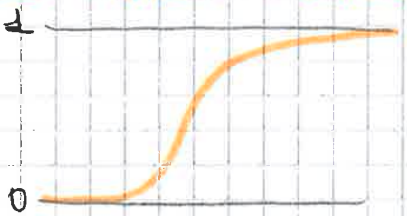
* CDF_cumulative DISTRIBUTION FUNCTION

ex: discrete DISTRIBUTION



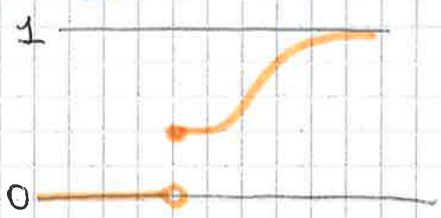
cumulative distribution function of a DISCRETE probability distribution

ex: continuous DISTRIBUTION



CONTINUOUS PROBABILITY DISTRIBUTION

ex:



example of a CDF of a distribution which has both a continuous part and a discrete part

We can compute quantities from PDF
We can summarize
We can compute quantities that give us the idea of how the distribution is
⇒ **Moments**

1. EXPECTATION: average value, mean, 1st moment (mean value of a DISTRIBUTION)

DISCRETE DISTRIBUTION: $E(x) = \sum_{i \in \Omega} x_i p(x_i)$

CONTINUOUS DISTRIBUTION: $E(x) = \int_{-\infty}^{+\infty} x p(x)\, dx$

2. VARIANCE: the spread, 2nd moment

DISCRETE DISTRIBUTION: $E(x) = \sum_{i \in \Omega} [x_i - E(x)]^2 p(x_i)$

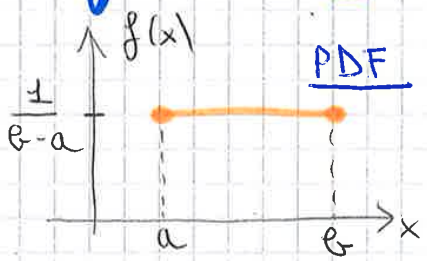
CONTINUOUS DISTRIBUTION: $E(x) = \int_{-\infty}^{+\infty} (x - E(x))^2 p(x)\, dx$

The sigma is used as a measure of the uncertainty of the expected value

When we do experiments (experiments several times) ⇒ we'll obtain number (the number every time is not the same) ⇒ those numbers are random variables from with we must be able to compute the expected value AND ALSO the variance
↳ gives the interval of uncertainty of the expected value
(it depends on HOW many times we repeat the experiment)

⚠ STATISCS IS based on PROBABILITY and HOW select DATA

Uniform Distribution



PDF $f(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{otherwise} \end{cases}$

This is a distribution that is very much used whenever we have to PUT some PRIOR AND there are not reasons to believe that one is more PROBABLE than ~~the other~~ one other

Bernoulli Random Variables

DEFINITION: Bernoulli Random Variable
Let X be a random variable taking only 2 possible values $\{0,1\}$.
Let $p = P(X=1)$.
Then X is said to be a Bernoulli random variable with parameter p, or equivalently: $X \sim Ber(p)$.

examples:
1. FLIPPING A "FAIR" COIN: $X \sim Ber\left(\frac{1}{2}\right)$
2. SUCCESS STARTING WINDOWS: $X \sim Ber(0.95)$

▶ Let $X \sim Ber(p)$. Then $\Rightarrow$

- $E[X] = 0 \times P(X=0) + 1 \times P(X=1) = p$ ← expected value for RANDOM Bernoulli variables
- $E[X^2] = 0^2 \times P(X=0) + 1^2 \times P(X=1) = p$
- $V(X) = E[X^2] - (E[X])^2 = p - p^2 = p(1-p)$ ← Variance of a Bernoulli Random Variables

Bernoulli Trials

- Independent Bernoulli random variables are important blocks to build more complicated random variables

$\Rightarrow$ Consider the following examples:

1. Flip a coin 10 times.
   Let X be the number of heads obtained
2. The packets received during a UDP connection arrive corrupted 5% of the times. Suppose you send 100 packets and let X be the number of packets received in error
3. In the next 30 births at an hospital, let X be the number of female babies.

$\Rightarrow$ These are all examples of Binomial random variables
(provided you assume each of the Bernoulli trials are independent)

Repeated Bernoulli Trials

Example: Suppose we send 3 packets through a communication channel. Assume
1. Each packet is received with a probability of 0.9
2. The events {packet 1 is received}, {packet 2 is received} and {packet 3 is received} are all INDEPENDENT
3. Let X be the total number of received packets

Then, we can build a table:

$X_i \sim Ber(p) \longrightarrow X_i = 1$ if {packet i was received}, $i = 1, 2, 3$

$X = X_1 + X_2 + X_3$

1 Exercise: Emily puts 60% of her free throws in basketball games. She had 25 free throws in last week's game. Use this information to answer the next 2 questions:

1. What is the average number of HITS?

$$E(x) = np = 25 \cdot 0{,}6 = 15$$

n: number of trials; p: probability of success

Everything we have faced up to now has 1 dimension.
Obviously we can have more variables → I can have A PROBABILITY DISTRIBUTION that DEPENDS ON MORE variables

Multivariate (Joint) Distribution

I have a variable X AND Y (RANDOM variables)

2D-CASE: $P(a \le X \le b, c \le Y \le d)$

dD-CASE: $P(a_1 \le X_1 \le b_1, \dots, a_d \le X_d \le b_d)$

We can generalize the above ideas from 1-dimension to any finite dimension

2D case → 2 variables that are discrete:

X = headache (MAL DI TESTA) (Y, N)
Y = flu (RAFFREDDORE) (Y, N)

"Y" → I have a headache; "N" → NO headache

ex: $P(\text{headache} \wedge \text{flu})$ = PROBABILITY to have a headache or a flu
$P(\text{headache AND NO flu})$ ... AND SO ON

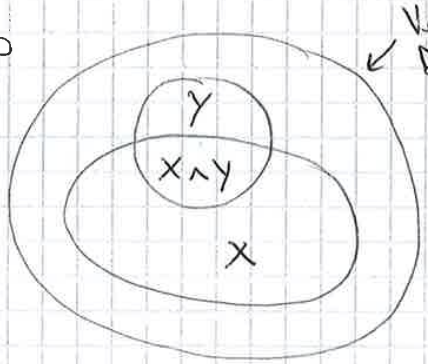
example: $P(\text{headache} \wedge \text{flu}) = 7/80$
$P(\text{headache, NO flu}) = 1/80$

$$\Rightarrow P(\text{headache}) = \frac{7}{80} + \frac{1}{80}$$

In general ⇒ TABLE:

	Flu	No Flu
headache	1/80	7/80
no headache	1/80	71/80

Venn Diagram



PROBABILITY distributions describe events which depend on one or more variables

dD-case ⚠

If I have $A \in \mathbb{R}^d$

- $P([X_1, \dots, X_d] \in A) = \int_A f(x_1, \dots, x_d)\, dx_1 \dots dx_d$
- $F_X(z_1, \dots, z_d) = \int_{-M}^{z_1} \cdots \int_{-M}^{z_d} f(x_1, \dots, x_d)\, dx_1 \dots dx_d$ multivariate CDF

exemple: London taxi drivers

A survey has pointed out a positive and significant correlation between the number of accidents and wearing coats.
Thay concluded that coats could hinder movements of drivers and be the cause of accidents.
A new law was prepared to prohibit drivers from wearing coats when driving.
Finally another study pointed out that people wear coats when it rains....

⚠ Be very carefull in reading DATA!!!
Because it's easy to do wrong → you mustn't try to impose your OPINION and you mustn't FIND DATA which support it because you will find these DATA but the models will be WRONG!

Formally: Conditional Independence
X IS CONDITIONALLY INDEPENDENT of Y given Z if:

$$P(x, y|z) = P(x|z)\,P(y|z)$$

exemple of TAXI DRIVERS IN LONDON:
$P(\text{Accidents}, \text{coats} \mid \text{RAIN}) = P(\text{Accidents}|\text{Rain})\,P(\text{Coats}|\text{Rain})$

⚠ $\forall(x,y,z) \quad P(X=x \mid Y=y, Z=z) = P(X=x \mid Z=z)$ ← equivalent to:

Chain Rule & Bayes Rule

CHAIN RULE:

- $P(x,y) = P(x|y)\cdot P(y) = P(y|x)\cdot P(x)$

from this equality, we get:

BAYES RULE:

- $$P(x|y) = \frac{P(y|x)\cdot P(x)}{P(y)}$$

exemple: AIDS test

Data that we have:
- APPROXIMATELY 0.1% of POPULATION are INFECTED
- TEST DETECTS ALL INFECTIONS
- TEST REPORTS POSITIVE for 1% healthy PEOPLE (FALSE POSITIVE)

RANDOM variable: $a = (\text{infected}, \text{non infected}) = (1, 0)$ (1 if INFECTED, 0 if NON infected)

$t = (\text{test}_+ \text{positive}, \text{test negative}) = (1, 0)$

$$P(a=1 \mid t=1) = \frac{P(t=1 \mid a=1)\,P(a=1)}{P(t=1)}$$

↳ PROBABILITY that a person is infected if the test has SAID POSITIVE

NAÏVE BAYES CLASSIFIER

- GIVEN: 1) The class PRIOR $P(Y)$
  2) ol CONDITIONALLY INDEPENDENT features (binary) $X_1, .., X_d$ given the class label Y
  3) For each X_i, we have the conditional likelihood $P(X_i|Y)$ → LIKELIHOOD
     ↳ I am assuming I am able to compute the conditional likelihood

Then ⇒ DECISION RULE:

$$f_{NB}(x) = \arg\max_y P(x_1 \dots x_d | y) \cdot P(y) = \arg\max_y \prod_{i=1}^{d} P(x_i|y) \cdot P(y)$$

↳ this is our first classifier → very simple
↳ AND in some cases it can be very effective

So, when AND HOW can we apply it when we have DISCRETE FEATURES?

NAÏVE BAYES ALGORITHM FOR DISCRETE FEATURES

We have TRAINING DATA that are:

$$\{(x^{(j)}, y^{(j)})\}_{j=1}^{n} \qquad x^{(j)} = (x_1^{(j)}, \dots, x_d^{(j)})$$

e.g. COLOR image → n samples
FOR INSTANCE, if I DO an OPTICAL character recognition

- d-dimensional features (ex: ARE images)
- n d-dimensional features + class labels
  ex: If I WANT TO classify APPLES, ORANGES, PEARS or other FRUITS... the label $j=1$ → APPLES, $j=2$ → ORANGES, $j=3$ → BANANA AND SO ON...

AND $x^{(j)}$ are all the images AND the PIXELS

$$f_{NB}(x) = \arg\max_y \prod_{i=1}^{ol} P(x_i|y) P(y)$$

we need to estimate these probabilities! ⚠

$\hat{P}$ ← HAT!
↳ APPROXIMATION 'cause it's an EXTIMATION

$$\hat{P}(y) = \frac{\{\#j : y^{(j)} = y\}}{n}$$

I approximate PROBABILITY with RELATIVE FREQUENCY

n = the total number of TRAINING DATA

I have n TRAINING DATA, and of these I obtain that ~~y training DATA are~~ A FRACTION of these n DATA are those that LABEL Y is equal "APPLE"

$\{\#j : y^{(j)} = y\}$ = number of times when j is such that $y^{(j)}$ is equal to a specific value

$$\hat{P}(y) = \frac{\{\#j : y^{(j)} = y\}}{n}$$

↳ Relative frequency

of these n TRAINING DATA, A FRACTION of these n I have that the label Y is equal to "APPLE" over the TOTAL number of TRAINING DATA that's the CLASS PRIOR for APPLE

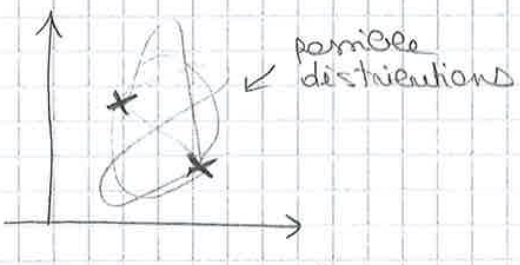
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 244

When we have DATA, we try to build a model of these DATA that explains the DATA.
A good model should be able to describe the phenomena and it should be able to predict the phenomena.

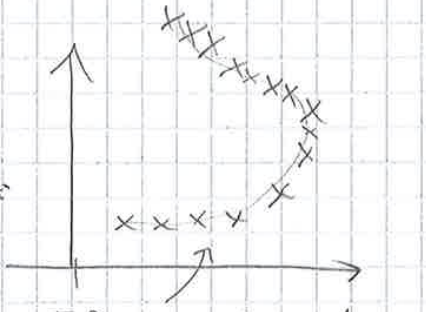
We need something that describes and predicts
then → challenges to our model

We need enough DATA!!

example: If I take 2 POINTS, what is the probability distribution?



⇒ If I sample a lot ⇒ A LOT of DATA



I have a clear idea of the DISTRIBUTION

FINO A Lezione ④

Parameter Estimation

FLIPPING A COIN

I have a COIN, if I flip it, what's the probability it will fall with the head up?

example: let us flip it a few times to estimate the probability

ex: head - tail - head - head - tail

ex ⇒ The estimated probability is $\frac{3}{5}$ ("Frequency of heads")

the outcome of the FLIPPING COIN is a random variable

Why? And How good is this estimation?
↳ of the real probability

MLE for Bernoulli distribution

ex: We have data, $D = \{head, tail, head, head, tail\}$

IN general $\boxed{\text{DATA } D = \{x_i\}_{i=1}^{M}}$, $\boxed{x_i \in \{H, T\}}$ → RANDOM variable

• M = number of times that I repeat the experiment

$x_i \in \{H, T\}$ → this RANDOM variable can have only 2 values

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i:x_i=H} \theta \cdot \prod_{i:x_i=T} (1-\theta) \longrightarrow \text{meaning of IID variables}$$

because the variables are identically DISTRIBUTED

↘ If I can estimate the probability for one, I can do for all

$$\boxed{\hat{\theta}_{MLE} = \arg\max_{\theta} \theta^{\alpha_H} \cdot (1-\theta)^{\alpha_T}}$$

- α_H = number of times the outcome is "Head"
- α_T = number of times the outcome is "Tail"

$$\boxed{J(\theta) = \theta^{\alpha_H} \cdot (1-\theta)^{\alpha_T}} \longrightarrow \text{I define a function } J \text{ which depends on } \theta$$

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \underbrace{\theta^{\alpha_H} \cdot (1-\theta)^{\alpha_T}}_{J(\theta)}$$

We want to find θ :

$$\frac{\partial J(\theta)}{\partial \theta} = \alpha_H \theta^{\alpha_H - 1} \cdot (1-\theta)^{\alpha_T} - \alpha_T \theta^{\alpha_H} (1-\theta)^{\alpha_T - 1} \Big|_{\theta = \hat{\theta}_{MLE}} \triangleq 0$$

we force $\emptyset$ to the 1st order derivative

$$\Rightarrow \alpha_H \theta^{\alpha_H - 1} - \alpha_T \theta^{\alpha_H} (1-\theta)^{-1} = 0$$

$$\alpha_H \theta^{-1} - \alpha_T (1-\theta)^{-1} = 0$$

$$\frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0 \Rightarrow \alpha_H (1-\theta) - \alpha_T \theta \Big|_{\theta = \hat{\theta}_{MLE}} = 0$$

$$\alpha_H - \alpha_H \theta - \alpha_T \theta \Big|_{\theta = \hat{\theta}_{MLE}} = 0$$

$$\Rightarrow \boxed{\hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}}$$

→ We have used the ML estimator principle to arrive to estimate that the MOST probable value of the probability of the RANDOM variable X being head is: $\frac{\alpha_H}{\alpha_H + \alpha_T} = \frac{\alpha_H}{n}$

So, the MAXIMUM LIKELIHOOD ESTIMATOR IS the FREQUENCY

What about prior Knowledge?

We know the coin is "CLOSE" to 50-50.
What can we do now?

Rather than extimate a single value for θ, we can OBTAIN a distribution over possible values of θ ⇒ THIS IS The Bayesan way

The Bayesan way...

RATHER than estimating a single θ, we can obtain a distribution over possible values of θ

- we DON'T GO FOR A single value, BUT we go for A DISTRIBUTION.

If the PRIOR is Beta DISTRIBUTION: → is our assumption of what we expect (our assumption of the problem) → A PRIORI → Based on what we know about the PROBLEM before the DATA

- $P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$

⇒ also the POSTERIOR will follow this DISTRIBUTION:
⇒ POSTERIOR is Beta DISTRIBUTION

- $P(\theta|D) \sim Beta(\beta_H + \alpha_H, \beta_T + \alpha_T)$

- $\hat{\theta}_{MAP} = \arg\max_\theta P(\theta|D) = \arg\max_\theta P(D|\theta) \cdot P(\theta)$ ← MAXIMUM LIKELIHOOD ESTIMATOR

$\hat{\theta}_{MAP}$ → MAXIMUM A POSTERIORI PROBABILITY

⚠ conjugate PRIOR: $P(\theta)$ AND $P(\theta|D)$ have the same form

$$\Rightarrow \hat{\theta}_{MAP} = \frac{\alpha_H + \beta_H - 1}{\alpha_H + \beta_H + \alpha_T + \beta_H - 2}$$

- We have to make 2 ASSUMPTIONS (it's really a lot)
  1. the probability of the data $P(D)$ is uniform
  2. the form of the DISTRIBUTION → $P(\theta)$

- $\hat{\theta}_{MAP} = \arg\max_\theta P(\theta|D) = \arg\max_\theta P(D|\theta)P(\theta) = \frac{\alpha_H + \beta_H - 1}{\alpha_H + \beta_H + \alpha_T + \beta_T - 2}$

MLE vs MAP

We've done all these exercizes because we want to estimate θ
→ I can do with the 1st derivative method → MLE → PURELY DATA BASED METHOD
OR → I can do with the PROBABILISTIC WAY (using the Bayes Rule) → which requires to improve my PRIOR Beliefs Believes → if what we know about the PROBLEM and our believes are very important (for example because we have a gerarchy or we have some dependencies) the Bayesian learning is the best way → it really works ~~even~~ expecially when we try to PUT a PRIOR, when we try to improve our believes

MLE: Maximum likelihood estimation
Choose value that maximizes the probability of observed data

$$\hat{\theta}_{MLE} = \arg\max_\theta P(D|\theta)$$

Hp: X_i are identically distributed and they follow the GAUSSIAN DISTRIBUTION

$$\Rightarrow \hat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i=1}^{m} \frac{1}{2\sigma^2} e^{-(X_i-\mu)^2/2\sigma^2}$$

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \left[\frac{1}{2\sigma^2} \exp\left[- \frac{\sum_{i=1}^{m} (x_i-\mu)^2}{2\sigma^2} \right] \right]$$

← I move the product into the argument of the exponential

- $\theta = (\mu, \sigma^2)$ → $= f(\theta)$

We get:

- $\hat{\mu}_{MLE} = \frac{1}{m} \sum_{i=1}^{m} x_i$
- $\sigma^2_{MLE} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \hat{\mu})^2$

⚠ σ^2_{MLE} → IT'S A BIASED ESTIMATOR

$$\sigma^2_{UN\,BIASED} = \frac{1}{m-1} \sum_{i=1}^{m} (x_i - \hat{\mu})^2$$ ← unbiased estimator

We say that this estimator is UNBIASED if the limit for the number of trials that we do (m) THAT goes to infinity the estimator goes to the real value

For the mean → we have that μ IS A GOOD estimator ~~even if BIASED~~ → IT'S UNBIASED
On the contrary, ~~use~~ for the variance, we have to ~~so~~ apply a SUBTRACT on m $(1/(m-1))$ → for $m \to \infty$, it goes to the real value

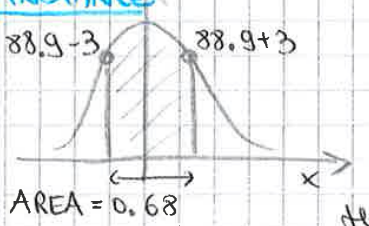
σ^2 IS A BIASED estimator
↳ we have to change $\frac{1}{m} \to \frac{1}{m-1}$ to have that at $m \to \infty$, σ^2 goes to the real value

⚠ Remark: MLE for the variance of a Gaussian IS BIASED
[the expected result of estimation is NOT the true parameter]

unbiased variance estimator: $\hat{\sigma}^2_{UNBIASED} = \frac{1}{m-1} \sum_{i=1}^{m} (x_i - \hat{\mu})^2$
we have to look to the UNBIASED ESTIMATOR of σ

- We have always to repeat the experiment several times if m IS BIG enough => GAUSSIAN DISTRIBUTION

FOR INSTANCE



ex: We have to report (after the experiments) the mean value: $\bar{x}$ and the variance
↳ $\bar{x} \pm \sigma$
FOR INSTANCE: 88.9 ± 3 $(\bar{x} = 88.9)$ $(\sigma = 3)$
We SAID that if I repeat the experiment another TIME, the PROBABILITY that my OUTCOME is between: 88.9-3 AND 88.9+3 => IS OF 70% (POOR PROBABILITY)

To demonstrate this, we need a LEMMA:

Tchebichev lemma

- I call: $\mu = \int x P(x)\,dx = \langle x \rangle$

$$\sigma^2 = \langle x^2 \rangle - \mu^2 = \int (x^2 - \mu^2) P(x)\,dx = \int (x-\mu)^2 P(x)\,dx$$

PROOVE:

$$(x-\mu)^2 = x^2 + \mu^2 - 2x\mu$$

$$\int (x^2 + \mu^2 - 2x\mu) P(x)\,dx$$

2μ → CONSTANT

$$-2\mu \int x P(x)\,dx = -2\mu^2$$

$\int x P(x)\,dx = \mu$ BY DEFINITION

$$\Rightarrow \int (x^2 - \mu^2) P(x)\,dx \quad (CVD)$$

- Let us suppose that σ^2 (variance) is finite

Then, the Tchebichev lemma says that:

$$P(|x-\mu| > t) < \left(\frac{\sigma}{t}\right)^2$$

$\mu \equiv$ mean of X

$\sigma \equiv$ variance of X

PROOF of the Tchebichev lemma:

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 P(x)\,dx > \int_{|x-\mu|>t}^{+\infty} (x-\mu)^2 P(x)\,dx$$

because I am integrating in a smaller interval

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 P(x)\,dx > \int_{|x-\mu|>t}^{+\infty} (x-\mu)^2 P(x)\,dx > t^2 \int_{|x-\mu|>t}^{+\infty} P(x)\,dx$$

because I am in the PART where $|x-\mu| > t$

$$\Rightarrow \int_{|x-\mu|>t}^{+\infty} P(x)\,dx < \frac{\sigma^2}{t^2} = \left(\frac{\sigma}{t}\right)^2$$

that is the Tchebichev lemma, demonstrated

NOW: PROOF of the LAW of LARGE numbers

1) Variance: $\mathrm{Var}\left(\sum_i X_i\right) = \sum_i V(X_i) \iff \{X_i\}$ are i.i.d → my RANDOM variables

$$\Rightarrow \mathrm{Var}(n\bar{X}_n) = n\sigma^2$$

$$\mathrm{Var}(n\bar{X}_n) = \mathrm{Var}\left(n \frac{1}{n} \sum_{i=1}^{n} X_i\right) = \sum_i V(X_i) \quad ?$$

2) $\mu(n\bar{X}_n) = n\mu$

⇒ Because of the Tchebichev lemma, we have:

$$P(n|\bar{X}_n - \mu| > t) < \left(\frac{\sigma^2 n}{t^2}\right)$$

Let me choose: $t = n\varepsilon$ $\Rightarrow P(n|\bar{X}_n - \mu| > n\varepsilon) = P(|\bar{X}_n - \mu| > \varepsilon) < \frac{\sigma^2 n}{n^2 \varepsilon^2}$

$$\Rightarrow P(|\bar{X}_n - \mu| > \varepsilon) < \left(\frac{\sigma^2}{n\varepsilon^2}\right)$$

USING DATA TO ESTIMATE PROBABILITIES

The Law of Large Numbers is what allows us to use data to estimate probabilities. ⇒ We MUST have enough DATA
↳ probabilities as frequencies

example: Suppose you stand in the bridge connecting the HG with the Auditorium and count the number of female and male individuals you see crossing the bridge for a period of 2 hours.

RANDOM VARIABLE $X_i = \begin{cases} \text{female} = "1" \\ \text{male} = "0" \end{cases}$

ex: We've counted ~~that~~ 283 different people and 27 of them were females

Let Y be the random variable representing be the gender of a randomly chosen individual inside TU/e.
Then it plausible that:

$$P(Y=1) \approx 27/283 = 0.0954$$

⇒ So, provided you can make the assumption that the individuals crossing the bridge are a representative independent and identically distributed sample of Y, we can use the LLN to estimate it's DISTRIBUTION

If we do the assumption that the random variables I am measuring are i.i.d., the frequency is the best approach we can have for the probability

The Central Limit Theorem

When discussing the normal distribution, we informally stated that the sum of independent random variables resembles a normal random variable.
Let's try to make this a bit more concrete:

Theorem:

Let $X_1, X_2, \dots, X_n$ be a sequence of independent and identically distributed (i.i.d.) random variables, with mean $\mu_X = E[X_i]$ and variance $V(X_i) = \sigma_X^2$

Define: $S_n = X_1 + X_2 + \dots + X_n = \sum_{i=1}^{n} X_i$ S_n = variable which is the sum of X_i random variables

Note that: $E[S] = \sum_{i=1}^{n} E[X_i] = n\mu_X$ and $V(S) = \sum_{i=1}^{n} V(X_i) = n\sigma_X^2$

Then, we have that the distribution of S is approximately normal, with mean $n\mu_X$ and variance $n\sigma_X^2$
That is:

$$S \sim N(n\mu_X, n\sigma_X^2)$$

The previous theorem was perhaps a bit sloppy (and technically not sound).
More formally, we have the following IMPORTANT theorem

$$\hat{\mu}_{MLE} = \arg\max_{\mu} \left(\frac{1}{2\delta^2} \exp\left(-\sum_{i=1}^{M} \frac{(x_i - \mu)^2}{2\delta^2} \right) \right)$$

$$J(\mu) = \exp\left(-\sum_{i=1}^{M} \frac{(x_i - \mu)^2}{2\delta^2} \right)$$

$$J_{LOG}(\mu) = \ln\left(\exp\left(-\sum_{i=1}^{M} \frac{(x_i - \mu)^2}{2\delta^2} \right) \right) = -\sum_{i=1}^{M} \frac{(x_i - \mu)^2}{2\delta^2}$$

$$\frac{\partial J_{LOG}(\mu)}{\partial \mu} = +\frac{1}{2\delta^2} \sum_{i=1}^{M} 2(x_i - \mu)\Big|_{\mu=\hat{\mu}} \triangleq 0 \quad \Rightarrow \quad \sum_{i=1}^{M} (x_i - \mu)\Big|_{\mu=\hat{\mu}} \triangleq 0$$

$$\sum_{i=1}^{M} (x_i - \mu) = \sum_{i=1}^{M} x_i - \sum_{i=1}^{M} \mu = \sum_{i=1}^{M} x_i - M\mu\Big|_{\mu=\hat{\mu}} \triangleq 0$$

$$\Rightarrow \hat{\mu}_{MLE} = \frac{1}{M} \sum_{i=1}^{M} x_i$$

OK, TORNA

Bayes Decision Rule

If I want to do a classification based on Bayes Decision Rule, I can say that:

- x is an observation for which:
  - if $P(\omega_1|x) > P(\omega_2|x)$ $\Rightarrow$ ω_1 is the true state of nature
  - if $P(\omega_1|x) < P(\omega_2|x)$ $\Rightarrow$ ω_2 is the true state of nature

[If I have a binary problem where I can have ω_1 or ω_2 = "true STATE of nature" means that this is TRUE]

PROBABILITY of error:

Therefore:
Whenever we observe a particular x, the probability of error is:

- $P(\text{error}|x) = P(\omega_1|x)$ if I decide ω_2
- $P(\text{error}|x) = P(\omega_2|x)$ if I decide ω_1

$P(\text{error}|x)$ = PROBABILITY that I'm MAKING an error given the observation x

When we use Bayes Rule to make decisions, we minimize the PROBABILITY of error

BAYES DECISION RULE MINIMIZES PROBAB. OF ERROR

↳ Decide ω_1 if $P(\omega_1|x) > P(\omega_2|x)$; otherwise decide ω_2

$\Rightarrow$ the PROBABILITY OF MAKING AN error given that I have observed x:

$$P(\text{error}|x) = \min\left[P(\omega_1|x), P(\omega_2|x) \right]$$

CONDITIONAL RISK:

I try to minimize the conditions of RISK:

- Minimizing $R \Leftrightarrow$ Minimizing $R(\alpha_i|x)$ for $i=1,\dots,a$

What is RISK? ← one of most common question at the exam!!!

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

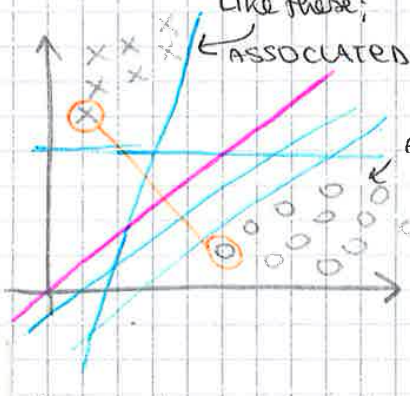
⚠ The RISK is the expected value for the loss, from the probabilistic point of view

- loss is a penalty of the decision (PREDICTION) and the action that I make
- If I change the LOSS, I change the algorithm (NATURE OF THE) !!!

~~It~~ It doesn't exist a one single loss!
The loss is a penalty
The loss is a DESIGNER CHOICE
If I change the LOSS, I change the behavior of the algorithm and I change dramatically the nature of the algorithm

Preview of next lessons:

We'll talk about 2-CLASS CLASSIFIERS
We'll have a situation where the DATA like these;



← ASSOCIATED with LABEL CLASS (1)

AND we'll have DATA like these associated with LABEL class (-1)

→ One of the 2-class classifiers is a PERCEPTRON *
↳ means FIND a LINE which separate the space IN 2 PARTS
IN one PART I WANT TO decide (1)
IN the other I WANT TO decide (-1)

Algorithm → We draw a line with dividers the space in 2 PARTS such that the distance between ~~the vectors between the closest test vector~~ the closest vectors is MAXIMUM

We'll find that the solution exists and it's unique → there is only 1 line

How can I get a different solution? Because the LOSS is different
If I define the LOSS differently ⇒ I obtain different solutions

I obtain only 1 line (the middle line) if I don't impose any penalty

02.6 23:00

It doesn't exist a good loss and a bad loss
Different losses bring to a different algorithm expectations with different properties

⚠ Once λ is defined, we can compute the RISK!
↳ expected value of the LOSS we have decided

⇒ decision is correct if $i=j$ and in error if $i \neq j$

We are always looking for a decision rule that minimizes the PROBABILITY OF ERROR which is the ERROR RATE

↳ So, we want to maximize the accuracy or to minimize the error rate. That is what we are going to do

[Error Rate = 1 − Recognition Rate]

↙ the lower it is, the happier we are

Model Selection

MAXIMUM LIKELIHOOD

↳ if we WANT A DATA DRIVEN APPROACH

If we have enough DATA, everything around work and we don't be worry about the RISK

If we want to do an estimation of what can be the PROBABILITY DENSITY FUNCTION we can suppose that the variables are i.i.d. and then estimate the PROBABILITY:

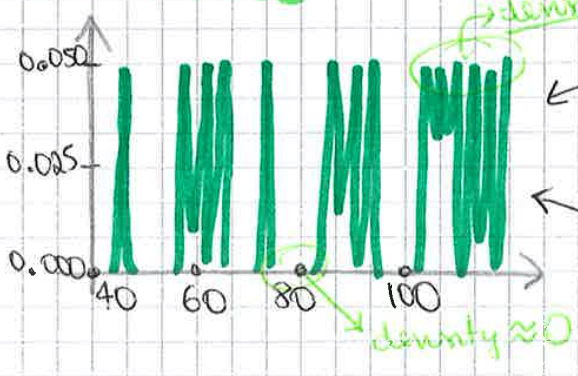
$$P_x\{X\} = \prod_{i=1}^{m} p(x_i)$$ PRODUCT of all the probabilities

MLE

- Need to measure how well we do
- For density estimation we care about: $P_x\{X\} = \prod_{i=1}^{m} p(x_i)$
- Finding a that maximizes $P(X)$ will peak at all data points since x_i explains x_i best...
- MAXIMA ARE DELTA FUNCTIONS ON DATA
- OVERFITTING!

example:
I have POINTS (my experimental POINTS)

OVERFITTING



likelihood on TRAINING SET is much higher than typical

I'm giving these training data and I want to maximize the probability of predicting these training data, what could I do ?

- I can put a delta function on each of my TRAINING DATUM.
  Delta function is defined as the probability = 1 for the closer of PROBABILITY Rule, I'm going to be fine → NO PROBLEMS

If I define a LOSS on my training data, formally I'm doing very well → BUT IT IS NOT GOING TO BE A VERY GOOD PREDICTOR because I'm never going to generalize
I'm never NO MISTAKES, BUT IT CAN'T BE A GOOD PREDICTOR

Model Selection

1) VALIDATION

- Use some of the DATA to estimate density
- Use other part to evaluate how well it works
- Pick the parameter that works best!

example: (non è IMPORTANTE)

$$\alpha(x'|x) := \frac{1}{M'} \sum_{i=1}^{M'} \log \hat{p}(x'_i)$$

→ There are 2 ways to DO MODEL SELECTION
THE FIRST IS called:

1) LEAVE-ONE-OUT CROSSVALIDATION

example: I have a classifier C which depends on DATA X AND on parameter α (whatever it is)

$C(x; \alpha)$ We have DATA: $x_1, x_2, \dots, x_{10}$
We know that α must be: $\alpha \in [0,1]$

1. At first I train $C(x_1 \dots x_9; \{0.1, 0.2, \dots, 0.9\})$ → I can decide many values for α HOW much is my COMPUTING POWER, I can estimate more and more fine this parameter

ONCE I have computed the TRAINING
2. NOW, I have to test the X_{10} DATUM
↳ the DATUM that I've cut

$C(x_1 \dots x_8 x_{10}, \{0.1 \dots 0.9\})$ → I substitute x_9 with x_{10}
then, I repeat it → I repeat with x_9 cut OUT

And I repeat it AGAIN AND the AGAIN

At each step I have to take note what is the best parameter α for that training

What is the BEST α among all the "BEST α_s" at each step that I do?
Majority rule
↳ for example $\alpha = 0.1$ for 3 times gives the best result
$\alpha = 0.3$ for 2 times // // // //
AND SO ON...
=> the BEST α is 0.1 (for instance)

Leave-one-out Crossvalidation

- Use ALMOST ALL the DATA to estimate density → whatever IS the FUNCTION WE WANT TO ESTIMATE
- Use single instance to ESTIMATE / TO TEST how well it WORKS
  ↳ one of our TRAINING DATA

$$\log p(x_i | X \setminus x_i) = \log \frac{1}{n-1} \sum_{j \neq i} K(x_i, x_j)$$

- This has huge variance
- Average over estimates for all training DATA
- Pick the parameter that works best

It's extremely costly → Very HIGH COMPUTATIONAL COST.

When we have the real parameters, the algorithm is stable
That's why the K-FOLD CROSSVALIDATION is really IMPORTANT

PCA - Principal Component Analysis

CONTENTS
1. MOTIVATION → We'll talk about MOTIVATION of this technique
2. PCA algorithms → We'll have 3 different algorithms for PCA
3. APPLICATIONS → examples of APPLICATION of these algorithms

1 Motivation

PCA is an UNSUPERVISED method
What does it mean "UNSUPERVISED"? ↓ It
I will get DATA AND I WILL NOT use a label associated to the DATA
I don't need a label
I get something information containing in these DATA come out thanks to the algorithm

It is used for
1. DATA VISUALIZATION
2. DATA COMPRESSION
3. NOISE REDUCTION

} these are the 3 main fields where PCA is used

1. DATA VISUALIZATION

→ 53 features

exemple: Given 53 BLOOD AND URINE samples (features) from 65 people
↳ People represent the samples
How can we visualize these DATA?
How can we visualize the measurements?

So, the (1) thing that I can do is → I have 65 samples → I can consider them as a vector

1) ⇒ MATRIX FORMAT (in the exemple: MATRIX 65×53)
↳ For sure ⇒ I look at all the DATA!

	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1							
A2							
A3							
A4							
A5							
A6							
A7							

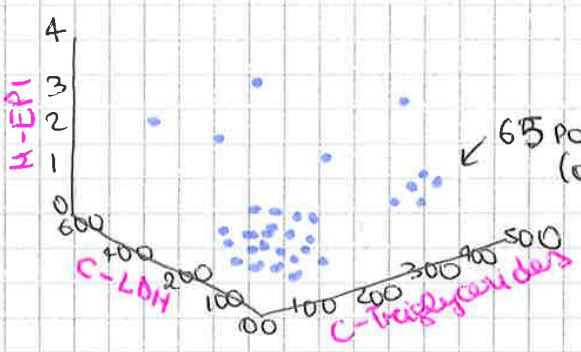
INSTANCES

features

I see all the numbers
Perceiving is not seen
↳ It's difficult to see the correlations between the features

This matrix doesn't really help the information to come out
What could we do?
Every instance is a person → every instance is a vector

5) So, maybe it's better to consider 3 features => Tri-Variate



This is not the solution

And this is just a simple example!
Actually, if we WANT TO do the same thing for an object recognition we have usually a lot of images and each image has a lot a lot a lot of pixels....
This is why the GRAPHS we've just seen DON'T WORK.

So, the question is:
Is there a representation better than the coordinate axes?
All of these representations are made in the DATA space that is given to us.
In the SPECIFIC CASE → we have 65 people and 53 features.
This is our DATA SPACE: 65×53
Maybe there is another space where the DATA are easier to understand
↳ maybe I can look for another space
on this space!

⇒ Is it really necessary to show all the 53 dimensions?
ex: If I am downloading an image on the web, in order to understand that that image contains a dog or a cat... DO I really need "1 million" of POINTS or I need less? PROBABLY much less

For the task at hand, for what I want to do, I need that dimension of the DATA or NOT? It will DEPEND ON that!
↳ for example: If I have to DISTINGUISH A DOG from A CAT, probably I need 5-10 POINTS
BUT If I have to DISTINGUISH 100 different types of DOGS ... probably I need much more POINTS !!!
↳ PROBABLY I need A bigger DIMENSION AND ALSO different DIMENSIONS

It depends always on what we have to DO!
It DEPENDS on our GOAL, on our TASK!

- ... what if there are strong correlations between the features?
- How could we FIND the smallest SUBSPACE of the 53-D space that keeps the MOST INFORMATION about the ORIGINAL DATA?

How we can FIND the smallest SUBSPACE...? the solution is: PCA

② PCA Algorithms (3 different types of ALGORITHMS)

Suppose that we have → these RED POINTS in a 2-dimensional space X_1 AND X_2

↓

LOOK at the GRAPH → next PAGE

lower dimensional space while preserving as much information as possible.

ex:
- Find best planar approximation to 3D data
- Find best 12-D approximation to 10^4-D data

and so on.....

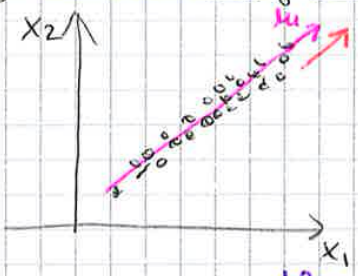
- In particular, choose projection that MINIMIZES squared error in reconstructing the original data

When we reconstruct the ORIGWAL DATA

1ST → change the reference system coordinate system of our data
2nd → APPROXIMATE our representation of the DATA
We can measure how much we are losing in terms of information since we've done an approximation

* Maximize the Variance

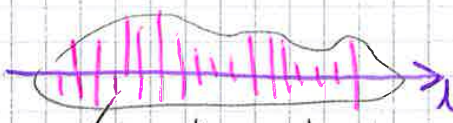
the variance is a measure of the spread of our DATA.
If the DATA vary a lot among one SPECIFIC DIRECTION



↳ I have to choose that DIRECTION as the new u_1 AXIS which maximizes the variance of the DATA
↳ the spread of the data in that direction is large → TO CAPTURE the MAJORITY of the INFORMATION OF these DATA

then, in particular, the u_1 AXIS is almost on the "centre" between the DATA because we WANT also to

* MINIMIZE the distances between the DATA AND the axis u_1
↳ or better, minimize the square of the distances



↳ in this situation we have that the DATA are varying A LOT along the u_1 AXIS ↓

this means that u_1 AXIS is able to capture a lot of informations of these data

PRINCIPAL COMPONENT ANALYSIS

- We obtain PCA vectors → they are originated from the center to the mean
given a lot of DATA, we can compute the centre of mass (centro di massa) of these data
AND then, we can put here the new zero of the new coordinate system

- PRINCIPAL COMPONENT number 1 → POINTS in the direction of the largest variance
⇒ it will be the POINT IN the DIRECTION of largest variance

- And then, depending on the dimension of the problem, we will have more principal components (more than 1)

We want to maximize the variance of the projections

$\underline{w}_1$ → vector such that when I take each sample vectors the projections with w_1, the square of these projections and then the sum of these squares is divided by the number of samples is maximized.

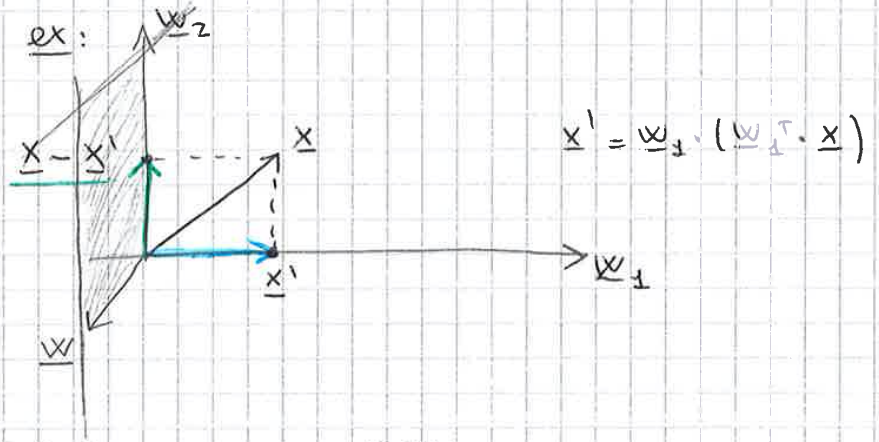
$$\underline{w}_1 = \underset{\|\underline{w}_1\|=1}{\arg\max} \frac{1}{m} \sum_{i=1}^{m} \left\{ (\underline{w}^T \cdot \underline{x}_i)^2 \right\}$$ ← this is the 1ST PRINCIPAL VECTOR

2nd PRINCIPAL VECTOR: $\underline{w}_2$

$$\underline{w}_2 = \underset{\|\underline{w}_2\|=1}{\arg\max} \frac{1}{m} \sum_{i=1}^{m} \left\{ \left[\underline{w}^T \cdot (\underline{x}_i - \underline{w}_1 \cdot \underline{w}_1^T \cdot \underline{x}_i) \right]^2 \right\}$$

$(\underline{w}_1 \cdot \underline{w}_1^T \cdot \underline{x}_i)$ is the PCA reconstruction → $\underline{x}'$

ex:



$$\underline{x}' = \underline{w}_1 \cdot (\underline{w}_1^T \cdot \underline{x})$$

We maximize the variance of the projection in the residual subspace

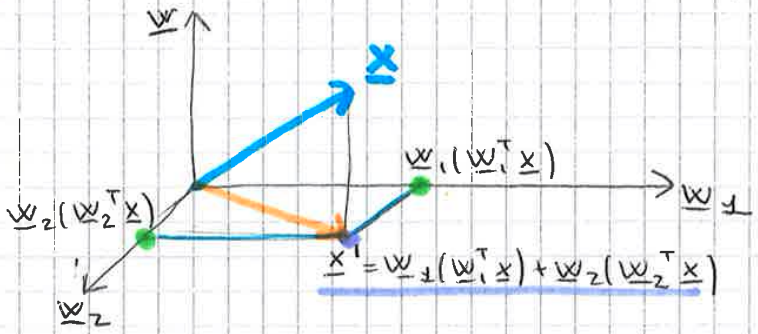
Now, we can generalize:

K-TH PRINCIPAL VECTOR: Given the $\underline{w}_1, \dots, \underline{w}_{k-1}$ we calculate the $\underline{w}_k$, K-th PRINCIPAL VECTOR:

$$\underline{w}_k = \underset{\|\underline{w}\|=1}{\arg\max} \frac{1}{m} \sum_{i=1}^{m} \left\{ \left[\underline{w}^T \left(\underline{x}_i - \sum_{j=1}^{k-1} \underline{w}_j \underline{w}_j^T \cdot \underline{x}_i \right) \right]^2 \right\}$$ K-th PCA vector

- $\underline{w}_k$: maximizes the variance of projection of $\underline{x}$
- $\underline{x}'$: PCA reconstruction in the space of $(k-1)$-dimensions

ex: in a 3-d SPACE: (first a 3D example)



We always WANT TO maximize the variance of the projection in what we called the RESIDUAL SUBSPACE

⇒ I will have: $\{\lambda_i, \underline{u}_i\}_{i=1\dots K}$ → they are the TOP PCA components

3) PCA algorithm III (SVD of the data matrix)

It's based on SVD → SINGULAR VALUE decomposition 1:03:58
↳ is a technique for decomposition of matrixes
↳ very efficient implementation

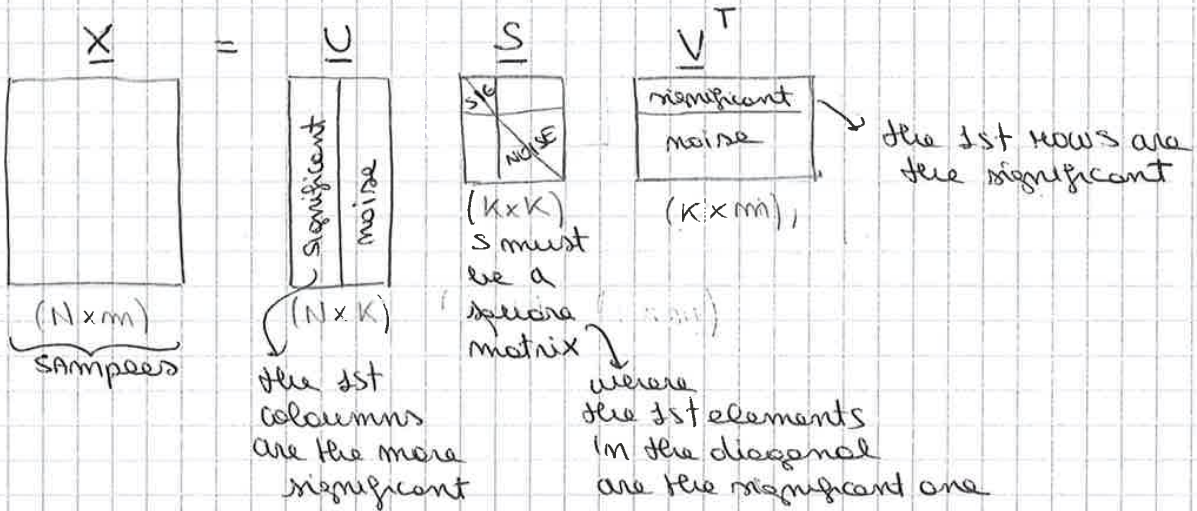
- We assume that we have DATA already centered on the centre of mass

Singular Value decomposition of the centered data matrix X

We have: $\underline{X} = [\underline{x}_1 \dots \underline{x}_m] \in \mathbb{R}^{N \times m}$ N = dimension of the data; m = number of examples/vectors

The matrix $\underline{X}$ of dimension: features × samples
it can be decomposed in:

$$\underline{X} = \underline{U}\,\underline{S}\,\underline{V}^T$$
features × samples



(*) See the book of Duda and Heart, "Pattern classification"

COLOUMNS of $\underline{U}$ → • they are the PRINCIPAL VECTORS: $\{\underline{u}^{(1)}, \dots, \underline{u}^{(K)}\}$
the FIRST is the MOST important
as we move to the left the vectors become less and less important

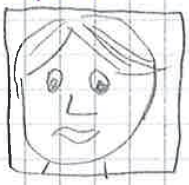
→ • they are ORTHOGONAL and with UNIT NORM
⇒ so: $\underline{U}^T\underline{U} = \mathbb{1}$ identity matrix
It means that $\underline{U}$ is a unity matrix

→ • we can reconstruct the DATA using a linear combination of $\{\underline{u}^{(1)}, \dots, \underline{u}^{(K)}\}$ → 1st K - PRINCIPAL VECTORS we can come back to the ORIGINAL DATA
this is always true ⚠

example: we can reconstruct images with very lower samples (?)

Applying PCA: Eigenfaces

example:



I have an image of face — DATASET ⇒ A LOT of images of faces

Each face is $\underline{x}$
- 256 × 256 values in PIXELS (luminance at location)
- $\underline{x} \in \mathbb{R}^{256 \times 256}$ (view as 64K dim vector)

We can see each image as a 64K-dimension vector

$\underline{X} = [x_1, \dots, x_m]$ (256 × 256 REAL VALUE; m faces)

We can find the COVARIANCE MATRIX

- FORM $\underline{X} = [x_1, \dots, x_m]$ centered DATA mtx
- COMPUTE the covariance matrix $\Sigma \rightarrow \Sigma = \underline{X}\underline{X}^T$

⚠ PROBLEM: By doing in this way, we get a covariance matrix Σ that is 64K × 64K !!! It's HUGE!

COMPUTATIONAL COMPLEXITY

- Suppose m instances, each of size N
  - ↳ examples: EIGENFACES → m = 500 faces
    - ↳ each face of size N = 64K
- GIVEN the N×N covariance matrix Σ, we can compute:
  - all N eigenvectors/eigenvalues in $O(N^3)$
    - ↳ to compute the eigenvalues, it has a complexity $O(N^3)$
  - the first K eigenvectors/eigenvalues in $O(KN^2)$
    - ↳ to compute the 1st k eigenvalues has the complexity $O(kN^2)$

BUT → if N = 64K ⇒ It's always very EXPENSIVE !!!

→ A clever workaround

- NOTE that: $m \ll 64K$ (FACES/)
  In the face recognition the number of instances (m) tends to be much smaller than the single vector dimension
- So, instead of the covariance matrix, it's probably better to do:

$$\underline{L} = \underline{X}^T\underline{X} \quad \text{instead of} \quad \Sigma = \underline{X}\underline{X}^T$$

$$\underbrace{\underline{X}^T\underline{X}}_{(m,64k)(64k,m)} = \underbrace{\underline{L}}_{(m,m)}$$

$\underline{X}$ = (256 × 256 REAL VALUES; m faces)

- $\underline{L}$ and Σ are related by the FACT that:

if $\underline{v}$ is an eigenvector of $\underline{L}$ ⇒ $\underline{X} \cdot \underline{v}$ is an eigenvector of Σ

Demonstration:

$\underline{L}\underline{v} = \gamma\underline{v}$ definition of eigenvector

$\underline{X}^T\underline{X}\underline{v} = \gamma\underline{v}$ I'm just using the definition of L

Then I multiply all by $\underline{X}$ on the left: $\underline{X} \cdot (\underline{X}^T\underline{X} \cdot \underline{v}) = \underline{X} \cdot (\gamma\underline{v}) = \gamma\, \underline{X} \cdot \underline{v}$

$$\Rightarrow \underline{X}(\underline{X}^T\underline{X}\underline{v}) = \gamma(\underline{X}\underline{v})$$

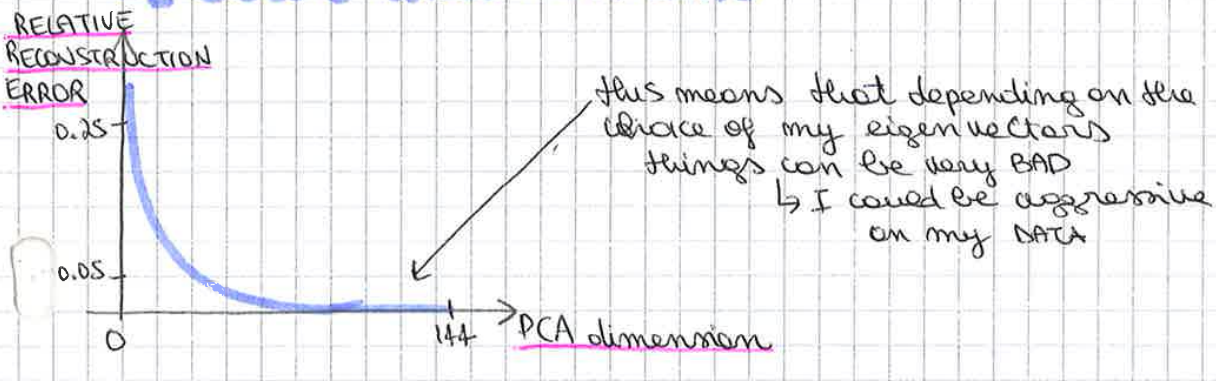
This method works very well if we are interested in a specific person recognition (e.g. Donald Trump security camera) or other cases....

2. Image Compression

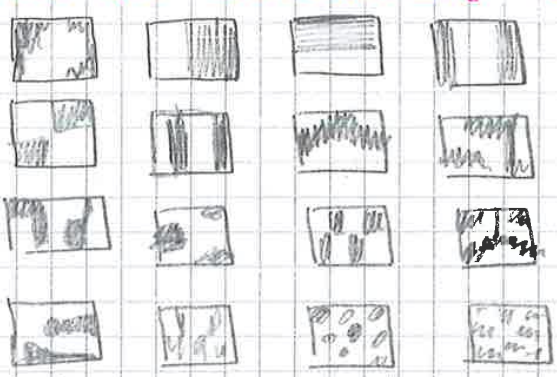
Dimensional reduction

example → ORIGINAL IMAGE → 372×492 PIXELS
- We can divide the original picture into PATCHES
  → EACH PATCH is an instance that contains 12×12 pixels on a grid
    ↳ view each PATCH as 144-dimension vector

L_2 error and PCA dim



- PAG 41 slide dez 9 → I can see the image reconstruction by reducing the PATCHES from 144 D to 60 D
- PAG 42 slide dez 9 → Reduction of the patches from 144 D to 16 D
  ⇒ the 16 MOST IMPORTANT eigenvectors are:



We have divided the image in localities
We are asking... what are the MOST recurrent locality in the global IMAGE?

- PAG 44 slide dez 9 → PCA compression: from 144 D to 6 D
  We start seeing that the image is strongly compressed
- PAG 46 slide dez 9 → FROM 144 D to 3 D
  It's not a "beautiful" image, but it's enough to recognize that there is a butterfly in the picture
- PAG 48 slide dez 9 → FROM 144 D to 1 D
  → it's A too STRONG compression
  Maybe it's too DIFFICULT to recognize the butterfly
  So → 3 D can be OK to recognize the SUBJECT

⚠ PAG. 49 slide dez 9 ⇒ the 60 MOST IMPORTANT EIGENVECTORS!
↳ they look like the discrete COSINE BASES of JPG!

PCA Conclusions:

PCA
- FINDS ORTHONORMAL BASIS FOR DATA
- SORTS DIMENSIONS IN ORDER OF "IMPORTANCE"
- DISCARD LOW SIGNIFICANCE DIMENSIONS

Uses: →
- TO GET COMPACT DESCRIPTION
- TO IGNORE NOISE
- TO IMPROVE CLASSIFICATION (hopefully!)

Not MAGIC because →
- It can't capture non-linear VARIATIONS
- It doesn't know CLASS LABELS

↓

which means that in classifications, expecially if we are very aggressive in reducing the dimensionality, we can obtain very bad surprises

PCA → is one of the many tricks to reduce dimensionality!
(it's not the only one)
↳ BUT it's the only that we'll do in this course

There are a lot of other methods for dimensionality reduction
↳ e.g.: Kernel PCA, PCA → ICA ... AND SO ON

Exercises on PCA:

1) The covariance matrix is given; $A = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$

GOAL: compute the PRINCIPAL VECTORS + associated eigenvalues

SVOLGIMENTO:

We have to compute the eigenvalues and the eigenvectors of the matrix A:

$$A\vec{v} = \lambda\vec{v}$$

$$(A - \lambda \mathbb{1}) = 0 \quad \Rightarrow \det(A - \lambda \mathbb{1}) = 0 \Rightarrow \det\begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix} = 0$$

$$\Rightarrow -(7-\lambda)(1+\lambda) - 9 = 0$$
$$-(7 + 7\lambda - \lambda - \lambda^2) - 9 = 0$$
$$-7 - 7\lambda + \lambda + \lambda^2 - 9 = 0$$

$$\lambda^2 - 6\lambda - 16 = 0$$

$$a\lambda^2 + b\lambda + c = 0$$

$$\lambda_{1,2} = \frac{6 \pm \sqrt{36 + 4 \cdot 16}}{2} \qquad \left[\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right]$$

$$\lambda_{1,2} = \frac{6 \pm \sqrt{100}}{2} = \frac{6 \pm 10}{2} \rightarrow \lambda_1 = 8, \quad \lambda_2 = -2$$

We've found the 2 eigenvalues
$\lambda_1 = 8$
$\lambda_2 = -2$

3) The given covariance MATRIX is: $A = \begin{bmatrix} 2 & 1 \\ -1 & 4 \end{bmatrix}$

Please, FIND the eigenvectors and the eigenvalues

$$\begin{vmatrix} 2-\lambda & 1 \\ -1 & 4-\lambda \end{vmatrix} = (2-\lambda)(4-\lambda) + 1 = 0$$

$$8 - 2\lambda - 4\lambda + \lambda^2 + 1 = 0$$

$$\lambda^2 - 6\lambda + 9 = 0 \rightarrow (\lambda - 3)^2 = 0$$

$$\lambda_1 = \lambda_2 = 3 \quad \text{OK}$$

$\lambda_1 = 3$

$\lambda_2 = 3$

$\Rightarrow$ the 2 eigenvectors will be identical $\underline{v}_1 = \underline{v}_2$

$$\begin{bmatrix} 2-3 & 1 \\ -1 & 4-\lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \begin{cases} -v_1 + v_2 = 0 \\ -v_1 - v_2 = 0 \end{cases} \Rightarrow v_2 = v_1 \Rightarrow v = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ OK}$$

What does it mean in terms of PCA?

→ my 2 DATA POINTS

→ it means that the 2 POINTS belong to the same straight line

⇒ I need only one dimension to represent these data (the second AXIS is zero)

I fit my POINTS in 1 line

4) The given Covariance matrix is:

$$A = \begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix}$$

Find the eigenvalues and the eigenvectors

$$\det \begin{bmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{bmatrix} = 0$$

$$\Rightarrow (1-\lambda)[(4-\lambda)(-5-\lambda) + 18] - 3(18 - 3(4-\lambda)) + 3(-18 - 6(-5-\lambda)) = 0$$

$$(1-\lambda)[-20 - 4\lambda + 5\lambda + \lambda^2 + 18] - 3(18 - 12 + 3\lambda) + 3(-18 + 30 + 6\lambda) = 0$$

$$(1-\lambda)[\lambda^2 + \lambda - 2] - 3(3\lambda + 6) + 3(6\lambda + 12) = 0$$

$$\lambda^2 + \lambda - 2 - \lambda^3 - \lambda^2 + 2\lambda - 9\lambda - 18 + 18\lambda + 36 = 0$$

$$-\lambda^3 + 12\lambda + 16 = 0$$

$$\lambda^3 - 12\lambda - 16 = 0 \rightarrow \quad (\lambda - 2)(\lambda^2 - 2\lambda - 8) = 0$$

$$\lambda^2 - 2\lambda - 8 = 0$$

$$\lambda = \frac{2 \pm \sqrt{4 + 32}}{2} \begin{cases} 4 & \lambda_1 = 4 \\ -2 & \lambda_2 = \lambda_3 = -2 \end{cases}$$

	1	0	-12	-16
-2		-2	4	16
	1	-2	-8	0

↳ $\lambda = -2$

+2

solution:

$\lambda_1 = 4 \rightarrow \underline{v}_1 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}$

$\lambda_2 = \lambda_3 = -2$

$\forall (\alpha_1, \alpha_2) \in \mathbb{R}^2 - \{0\}$

↳ $\underline{v}_2 = \underline{v}_3 = \alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad \forall \alpha_1 \; \forall \alpha_2$

Perceptron

(our 1st and only machine algorithm of this course)

The perceptron is the very grand-grand father of deep learning. This is because Perceptron was the 1st algorithm that it HAS been explained by the functions of the brain (?)

- EARLY theories of the BRAIN
  - ↳ we try to MIMIC the BRAIN learning DATA and so PROBLEMS (?)

BIOLOGY AND LEARNING

Basic Idea inside the PERCEPTRON:

- GOOD Behavior should be rewarded, bad behavior should be punished (or not rewarded). This improves system fitness
  - ↳ ex: yesterday, my dog decided to express itself on the carpet. It hasn't been rewarded. If my DOG brings me the ball, when I ask him... I give him a candy,.... AND SO ON

We have to have correlated actions → ABILITY to generalize
↳ "set" of BAD ACTIONS AND GOOD ACTIONS correlated

- KILLING a saber-tooth tiger should be rewarded ...
- CORRELATED events should be combined
- PAVLOV'S SALIVATING DOG

ex: if it's not OK in a part of the carpet, it's not ok also in all the carpet

This is actually a very strong mechanism for learning in BIOLOGICAL systems and for HUMANS
↳ Mother who hurries up with her child to educate him....

TRAINING MECHANISMS:

- Behavioral modification of individuals (Learning)
  - ↳ SUCCESSFUL behavior IS REWARDED (e.g. when I give food to my DOG)
- HARD-CODED behavior in the genes (INSTINCT)
  - The WHOLE genetically coded animal does not reproduce

We are not tabula RASA. We're not learning from scratch

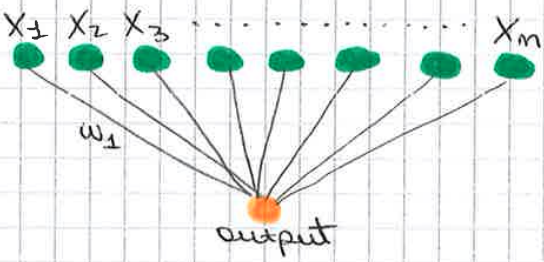
It's a combination in the case of biological systems of priors that are INBUILT coded in DNA → we are not TABULA RASA
When we were born, we are biological older than our parents because we have all their DNA, all their experiences IN US, IN the genetic encoding
- ↳ e.g. GRANDPARENTS have difficulties in learning how a smartphone works, while a child finds it very easy
- ↳ this is not mimicked (HOWEVER) very well by the Perceptron, (?) while Deep Learning (MECHANISM) represents quite well this fact

And I also WANT TO BE invariant with respect to small VARIATIONS, like for instances TRANSLATIONS
→ What I could do?

Perceptron

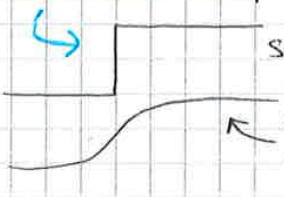
- **weighted linear combination**
  - ↳ where I can put a BIAS → "b" ↓ ADD A CONSTANS
  - which makes invariant from a TRANSLATIONS

$$f(x) = \sigma(\langle w, x \rangle + b)$$



↳ then I use a σ (SIGNUM FUNCTION) which takes into account the threeshold:

if $(\langle w, x \rangle + b)$ is above the threeshold, it is FALSE
otherwise NOT



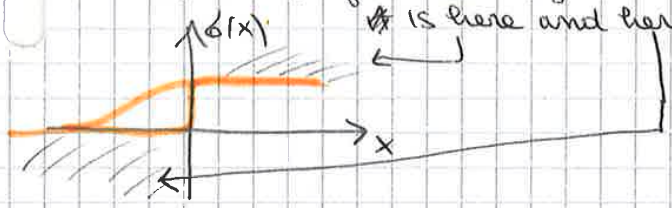
something like A STEP FUNCTION

OR better (more realistically) smoother than a step function ←

- WE WANT A DECISION FUNCTION which is NOT linear
  ⇒ **Non linear decision FUNCTION** → that's because we've put the SIGNUM FUNCTION
- **linear separating hyperplanes** (SPAM / HAM, NOVEL / TYPICAL, CLICK / NO CLICK) → this is done by adding a BIAS
  ↳ I can decide if this w, with x, it actually defines a plane

(*) BOOK: Cristianini, Shawe, Taylor "An Introduction to SVM"

I am dividing the world from what # is here and here, basically.



Where we have a decision function like this, what we are really building is a classifier that finds a linear separating hyperplanes

I am interested in BINARY PROBLEMS
And I'm trying to FIND how to separate them
The quantities I am interested in are w_i AND b ↳ BIAS
(x_i are the INPUTS)

it is going to tell me where the hyperplane IN two dimension IS ALIENED (above space, BOVESTA)
↳ if IT PASSES from the ORIGIN OR NOT, BASICALLY

▶ What do we learn from the Perceptron?
LEARNING means: ESTIMATING the PARAMETERS w AND b

Once I know those (w AND b), I know my classifier

If I substitute: $w \leftarrow w + y_i x_i$
$b \leftarrow b + y_i$

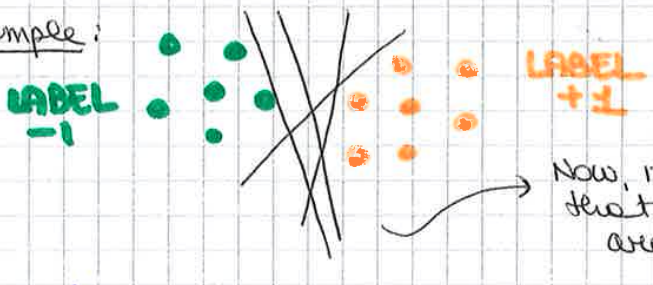
↳ here ⇒ $y_i[\langle w, x_i \rangle + b]$ ⇒ this becomes POSITIVE

So, HOW DOES THIS ALGORITHM WORK?

- I take my TRAINING DATA
- I define the equation of a plane in a n-dimensional space
- I want to FIND the value of the vector which defines the slope and the constant that gives the intercepts on my axis (AXIS of the plane), such that it separates correctly all my training DATA

↙
correctly, in my example, means that all the training DATA having label $y_i = -1$ is in one of the side of the plane and all the data having the other label in in the other side of the plane

example:



Now, it's easy to understand that the possible solutions are infinite

Algorithm:

- initialize $w=0$ and $b=0$
- repeat
  - if $y_i[\langle w, x_i \rangle + b] \leq 0$ then
    $w \leftarrow w + y_i x_i$
    $b \leftarrow b + y_i$
  - end if
- until all are classified correctly

↳ IS a linear combination:

* Remember that the weight vector w is: $w = \sum_{i \in I} y_i x_i$

So, I have my DATA that arrive:

⇒ • NOTHING HAPPENS IF CLASSIFIED CORRECTLY
IF a training DATUM arrives for which the classification is correct, nothing happens
If something arrives which is classified wrong, I will update

* How it is design this algorithm, it's intrinsically online because it updates his decision function (w AND b) step by step
↳ I can have frames that arrives from a camera } examples
↳ the frame arrives step by step

↙
this is what we call online learning
↳ it means that I don't have all the training data that have to arrive all together until I have finished learning

IN THIS ALGORITHM I build my classifier on the FLY STEP BY STEP ↳ the more data I have, the more experiences I do, the better I can understand of the problem

Convergence Theorem

Theorem:

- If there exists some values (w^*, θ^*) with unit length and

$$y_i [\langle x_i, w^* \rangle + \theta^*] \geq \rho \quad \forall i$$

↳ A certain value ρ

- ⇒ Then the perceptron converges to a linear separator after a number of steps bounded by $(\theta^{*2}+1)\cdot(r^2+1)\,\rho^{-2}$ where we have that: $\|x_i\| < r$

Remark:

- (w^*, θ^*) = values at convergence

What the theorem PRACTICALLY SAYS:

IF IT POSSIBLE TO FIND an hyperplane which separates exactly the 2 classes of training DATA ⇒ the algorithm, that we have described before, is going to FIND that hyperplane AND it will FIND IT IN A FINITE NUMBER OF STEPS:

Then, if we know that the solutions are linearly separable, then we garantee the convergence of the algorithm

Remark: Note that there is nothing here about the dimension of the DATA

- DIMENSIONALITY INDEPENDENT

↳ If the problem is variable separable, it doesn't matter if it is 2D, 3D, ..., MD ... the algorithm will always FIND a solution

Remark:

- ORDER INDEPENDENT → The theorem DOES NOT DEPEND ON the order in which I give the DATA

The solution that I find here is not "OPTIMAL" → whatever it means
↳ it's a possible solutions of many

- Even though the solution I will find, it depends on the order of DATA !!

↙ Because my training DATA are: $\{(x_i, y_i)\}_{i=1}^{m}$

BUT my w are DEFINED in another set of DATA: $w = \sum_{i \in I} y_i x_i$

↳ AND thus SUBSET depends on all the i for which I made a mistake (for which the sign of the DECISION function was negative)

↳ SUBSET OF TRAINING DATA

If I change the order of the DATA ⇒ I will take different SUBSETS ⇒ the weight vector changes as a consequence!

⚠ The solution we will FIND depends on the order of the DATA!

This theorem doesn't depend on the order of the DATA

Remark:

- It is going to scale with respect to how the PROBLEM is difficult
  ↳ the SCALING is made by the CONSTANTS (?)

Alignment → defined as:

$$\langle (w_i, \theta_i), (w^*, \theta^*) \rangle$$

We WANT TO PUT A BOUND on the number of steps to converge
I know by definition of the theorem .... I said that when I am in w^* and θ^* I am a δ-convergence (?)
I WANT TO KNOW when I am at step i, how far I am from (w^*, θ^*) ?
↳ and w AND θ are → (w_i, θ_i)

⇒ scalar product: $\langle (w_i, \theta_i), (w^*, \theta^*) \rangle$
↳ one way to measure it (SCALAR PRODUCT between 2 vectors)

If I make a mistacke → I have to update !!

▶ If I make an error on the sople (x_i, y_i), I need to update w

$$\Rightarrow \langle (w_{j+1}, \theta_{j+1}), (w^*, \theta^*) \rangle =$$

$$= \langle [(w_j, \theta_j) + y_i (x_i, 1)], (w^*, \theta^*) \rangle$$

$$= \langle (w_j, \theta_j), (w^*, \theta^*) \rangle + y_i \langle (x_i, 1), (w^*, \theta^*) \rangle$$

$$\geq \langle (w_j, \theta_j), (w^*, \theta^*) \rangle + \rho \geq j \cdot \rho$$

↳ because ρ is the closest ⇒ is the MINIMUM → $\rho = \min_i y_i \langle w, x_i \rangle$

The alignment increases with the number of errors → j
The more mistakes I make, the more the alignment increases

$$\boxed{\langle (w_{j+1}, \theta_{j+1}), (w^*, \theta^*) \rangle \geq j\rho} \rightarrow \text{result of STEP 1}$$

STEP 2: We take this quantity:

$\langle (w_{j+1}, \theta_{j+1}), (w^*, \theta^*) \rangle$ AND we use the Cauchy-Schwartz inequality for the DOT product:

$$\rightarrow \langle (w_{j+1}, \theta_{j+1}), (w^*, \theta^*) \rangle \leq \| (w_{j+1}, \theta_{j+1}) \| \cdot \| (w^*, \theta^*) \|$$

(≤: for the Cauchy-Schwartz inequality)

NORM of the NORMAL vector which defines the hyperplane at convergence (with a given DIMENSION)
[↳ if w^* is of DIMENSION D ⇒ (w^*, θ^*) has DIMENSION $(D+1)$]

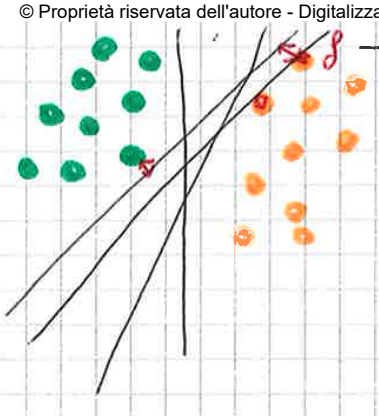
$$\Rightarrow \| (w^*, \theta^*) \| = \sqrt{1 + (\theta^*)^2}$$

(θ^*: IS A CONSTANT)

w^* is the normal vector ⇒ $\| w^* \| = 1$

$$\Rightarrow \boxed{\langle (w_{j+1}, \theta_{j+1}), (w^*, \theta^*) \rangle \leq \sqrt{1 + (\theta^*)^2}\, \| (w_{j+1}, \theta_{j+1}) \|} \leftarrow \text{result of STEP 2}$$

STEP 3: Note that we have FOUND an upper and a lower bound for the ALIGNMENT



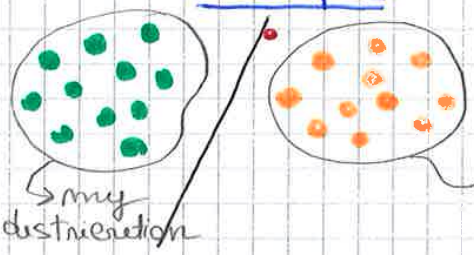
→ In these possible different solutions → the ρ changes

ρ → MARGIN

Remember that we have said that all the POINTS that we are taking, they are coming from a PROBABILITY DISTRIBUTION

↳ H will be live in a N-dimensional space

example:

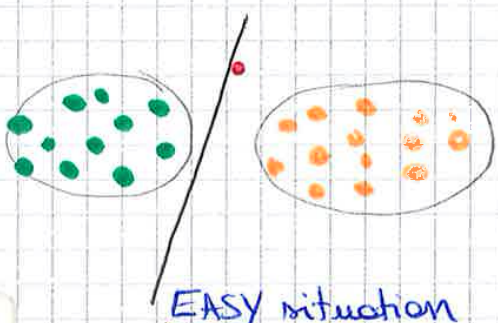


my distribution

FOR instance I can say that the PROBABILITY DISTRIBUTION IS something like this

Is it possible that a new data arrives in THIS POINT • for instance?
Yes, it's not impossible because for instance it can arrive from the tail of the probability distribution

examples:



EASY situation

in this example I think things to be easy
(but I can anyway DO mistake → FOR INSTANCE this POINT •)

it's a harder PROBLEM

HARD

In this example where I have a very small MARGIN

↳ OF course, I still FIND a solution
I still CAN FIND many solutions
BUT with less degree of freedom

- So, MARGIN vs SIZE makes a problem less or more hard

Concepts & Version space

All of these reasoning can be extended to spaces that are not linear (for example NON euclidian spaces)

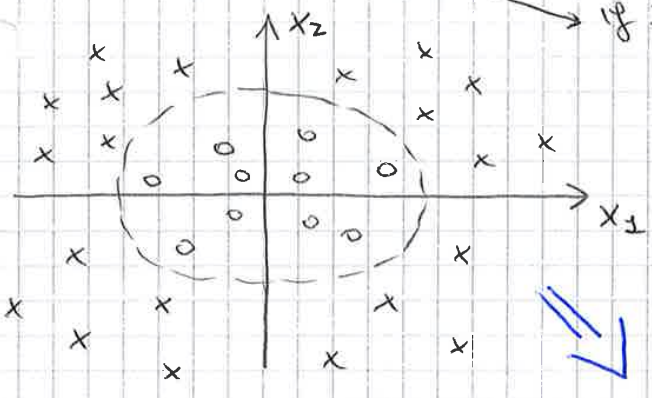
All the problems we have considered are linearly separable

→ If I am in a non linear space, then the variable are not separable

Challenges: OK, my PROBLEM is not variable separable. But If I would know ~~HOW~~ some informations to make mistake
→ maybe I still could find a solution

for instance: **QUADRATIC FEATURES**

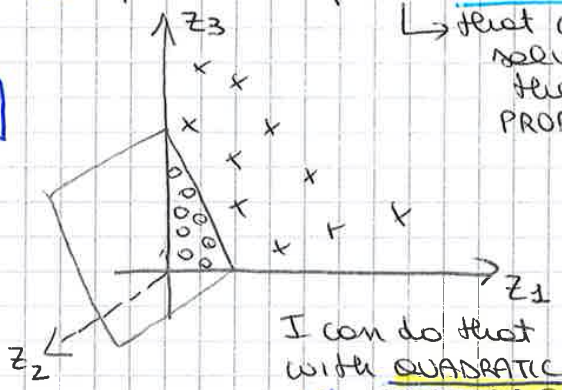
ex: let's say that this is the problem that we have:



→ if I could build a classifier which is an ellipse → It would be FINE

But all I can do are planes!

- So, I can map in a 3D SPACE:
  → that actually solves the PROBLEM



Don't worry about the functional form of these features

I can do that with QUADRATIC FEATURES

- SEPARATING SURFACES are: CIRCLES, HYPERBOLAE, PARABOLAE

⚠ If I live in a space where the data that are given are not linearly separable, I could always think of mapping into an higher dimensional SPACE WITH A NON LINEAR MAPPING
And my hope is that by changing my space, in that space my problem might become linearly independent

* this is called the **Kernel Trick** → very powerful trick
  → it is also APPLICABLE on PCA
  → Kernel PCA

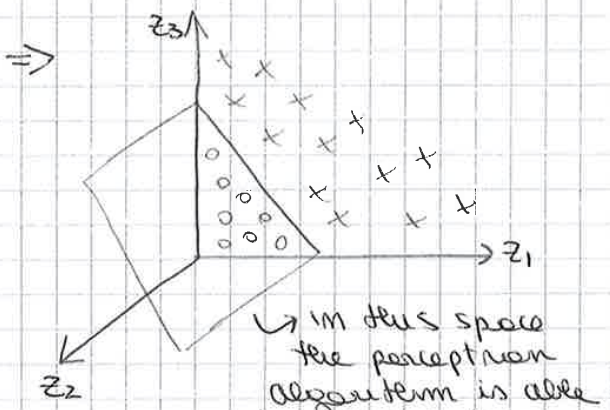
trick: if we have BUILT our HYPERPLANE in our space where the PROBLEM IS linearly separable

It's a way to change the metric of the space

AND then we DO the MAPPING INTO higher DIMENSIONAL SPACE => WE OBTAIN SOMETHING NON LINEAR

* What really changes things for us is
  - DATA we have
  - LOSS
  - metric ← ⚠ very IMPORTANT

example:



↳ in this space the perceptron algorithm is able to build the hyperplane which separates correctly the algorithm

This can work with many different surfaces
↳ circles, hyperboles, ... AND SO ON

So, we don't need separating surfaces to be linear.
They could be circles, hyperboles, parabolas.... not linear at all

The trick of using classifier that delievers linear solutions seems to be find the right non linear transformation of our data such that the problem becomes linearly separable

this is the conceptual mapping we are doing

So, we need to find features to transform our problem from non-linearly separable to linearly separable

↳ **CONSTRUCTING FEATURES** (Very manual OCR system)

ex: Let's say that I need to build an optical character recognition system
I WANT TO RECOGNIZE DIGITS: 1,2,3,4,5,6,7,8,9,0
because I WANT TO BUILD the system for the BANK for cashing automatically a checks which EXISTS by the way

OCR → OPTICAL character Recognition

Let's look at the DATA:

	1	2	3	4	5	6	7	8	9	0
loops	0	0	0	1	0	1	0	2	1	1
3 joints	0	0	0	0	0	1	0	0	1	0
4 joints	0	0	0	1	0	0	0	1	0	0
Angles	0	1	1	1	1	0	1	0	0	0
Ink	1	2	2	2	2	2	1	3	2	2

When I write a digit sometimes I have loops, sometimes I have angles....

ex: when I write '8' I have 2 LOOPS
when I write '6' I have 1 LOOPS
.....

→ What type of INK IS GOING TO BE USED?

Feature Engineering for Spam Filtering

If I WANT TO DO a SPAM FILTERING
PROBABLY I have to LOOK AT:

- BAG OF WORDS
- PAIRS OF WORDS
- DATE & TIME
- RECIPIENT PATH
- IP NUMBER
- SENDER → is a inside the list of allowed contacts or not
- ENCODING
- LINKS
- ... SECRET source ...

Again, this is going TO NOT WORK... (?)

So, we can write the classifier as a function of the scalar product in the space defined by the mapping ϕ

⚠ Whenever that happens to you for classifier, you are officially HAPPY → it is an extremely important property of an algorithm

⚠ Whenever we can express something in terms of scalar products in a certain space, we have something in our hands → it gives us very powerful tools

So ↓

- Classifier is linear combination of inner products:

$$f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$$

▶ Now, we know how compute NON-linear Perceptron (let's say: Perceptron on features)
We still don't know how to define these features → FUNCTION ϕ
⇒ let's introduce Kernels

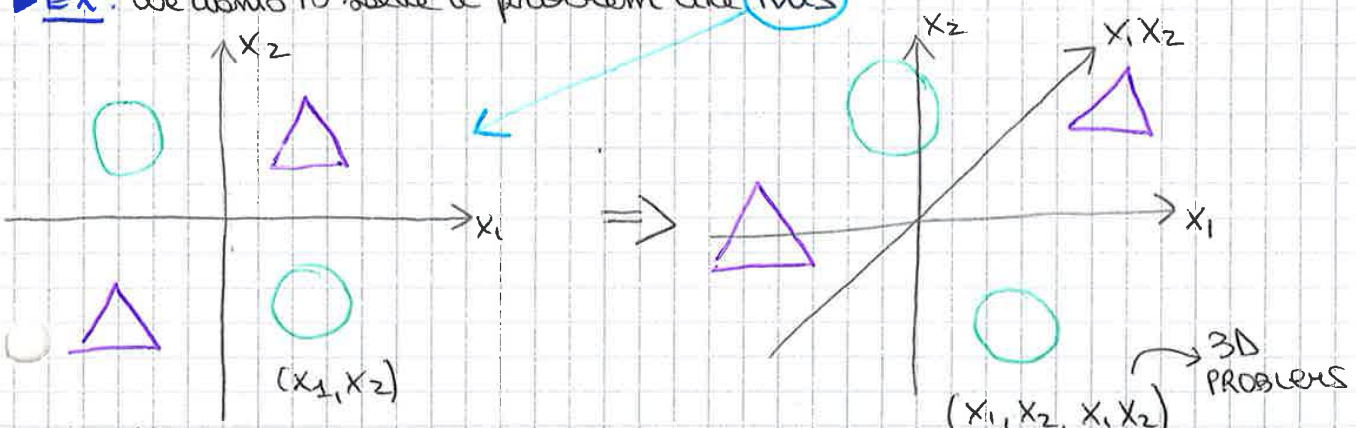
- As we've said → we have PROBLEMS → because features by HANDS DON'T WORK ⇒ SO: Kernels — WE WANT TO SOLVE IT with Kernels

PROBLEMS

- PROBLEMS
  - Need DOMAIN EXPERT (e.g. Chinese OCR)
  - Often expensive to compute
  - Difficult to transfer engineering knowledge
- SHOTGUN SOLUTION
  - Compute many features
  - Hope that this contains good ones
  - Do this efficiently

Kernels

▶ EX: We want to solve a problem like this



- XOR not linearly separable
- MAPPING INTO 3 dimensions makes it easily solvable

- example: second order features in 1000 dimensions.
  This leads to:
  $5 \cdot 10^5$ numbers
  For higher order polynomial features ⇒ MUCH WORSE

- SOLUTION ⇒ Don't compute the features, try to compute dot PRODUCTS implicitely.
  For some features this works....

If I can DEFINE a good SCALAR PRODUCT (non linear) → this is all what we need
→ this is the Kernels trick

Whenever I can write an algorithm as a function of the scalar product → Then, we can always make this algorithm not linear, by substituting the scalar product with a Kernels function that represents the scalar product in some higher dimensional space which I don't know what it is and I don't care about what it is
I just know the MAP from a lower dimensional space to an higher one by a non linear mapping

This is extremely powerful

Kernelizing → making non linear an algorithm, is very powerful

DEFINITION of KERNEL FUNCTION

Def: A Kernel Function $K: \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ is a symmetric function in it's arguments
for which the following property holds:

$K(x, x') = \langle \phi(x), \phi(x') \rangle$ for some feature map ϕ

So, we have done the mapping from a lower dimensional space to an higher one
And we image to be able to define a function K such that we can express the algorithm as a function of the scalar product

Does the Kernels K define univocally (UNIVOCAMENTE) ϕ?
Or do we have a family of ϕ?

The answer is NO → the possible mappings to an higher dimensional space are infinite, actually

(This is obviously done if $K(x, x')$ is much cheaper to compute than $\phi(x)$ ....)

The Kernel Perceptron

From the features Perceptron, we are now reading to go toward the Kernel Perceptron

- So, what do we WANT FROM Kernel functions? What type of property?

→ Kernel Conditions

1. COMPUTABILITY
We have to be able to compute $K(x, x')$ efficiently (much cheaper than dot products themselves)
↳ Obviously we WANT TO make our life easier by using Kernels AND not more complicated

2. "NICE AND USEFUL" functions
The features themselves have to be useful for the learning problem at hand.
[Quite often this means smooth functions.
↳ The idea is to use Kernels instead of the scalar products my algorithm get better → better performance
↳ faster convergence
---- MORE powerful tools

We should have: $K(x, x') = K(x', x)$ symmetric property
↳ Because the scalar product is symmetric by construction:
$\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$

3. SYMMETRY
Obviously: $K(x, x') = K(x', x)$ due to the symmetry of the DOT PRODUCT:
$\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$

But this is not enough → Having a symmetric function on 2 variable is not enough to say that it will exist always a mapping function Φ s.t. K is really A DOT PRODUCT

DOT Product in Feature Space
Is there always a Φ such that K really is a DOT PRODUCT?

We have a Theorem

Mercer's Theorem

Theorem:

For any symmetric function $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is square integrable in $\mathcal{X} \times \mathcal{X}$ and that satisfies:

$$\int_{\mathcal{X} \times \mathcal{X}} K(x, x') f(x) f(x') \, dx \, dx' \geq 0 \quad \forall f \in L^2(\mathcal{X})$$

Then, there exists A MAPPING: $\Phi_i : \mathcal{X} \to \mathbb{R}$ and $\lambda_i \geq 0$ where:

$$K(x, x') = \sum_i \lambda_i \Phi_i(x) \Phi_i(x') \quad \forall (x, x') \in \mathcal{X} \times \mathcal{X}$$

- From Kernel functions, we can derive **Kernel Matrix**

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = K(x_i, x_j)$$

where: $x_i \longrightarrow$ TRAINING DATA

I can build the Kernel Matrix based on my training DATA and then derive its properties

↑ IF I know the Kernel function

Kernel Matrix

To compare observations we compute dot products, so we study the matrix K given by:

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = K(x_i, x_j)$$

where x_i are the TRAWING PATTERN

each scalar product is a similarity measure

• SIMILARITY MEASURE

The entries K_{ij} tell us the overlap between $\phi(x_i)$ and $\phi(x_j)$
So, $K(x_i, x_j)$ is a similarity measure

• K IS POSITIVE (SEMI) DEFINITE

▶ Because of K IS POSITIVE SEMIDEFINITE, we can make a CLAIM:

CLAIM: K IS POSITIVE SEMIDEFINITE

- $\alpha^T K \alpha \geq 0 \quad \forall \alpha \in \mathbb{R}^m \quad \forall$ Kernel matrices $K \in \mathbb{R}^{m \times m}$

PROOF:

$$\sum_{i,j}^{m} \alpha_i \alpha_j K_{ij} = \sum_{i,j}^{m} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle =$$
$$= \left\langle \sum_{i}^{m} \alpha_i \phi(x_i), \sum_{j}^{m} \alpha_j \phi(x_j) \right\rangle = \left\| \sum_{i=1}^{m} \alpha_i \phi(x_i) \right\|^2 \geq 0$$

▶ We can talk about: **Kernel EXPANSION**:

If w is given by a linear combination of $\phi(x_i)$, we get:
[which is the case of PERCEPTRON]

$$\Rightarrow \langle w, \phi(x) \rangle = \left\langle \sum_i \alpha_i \phi(x_i), \phi(x) \right\rangle = \sum_i \alpha_i K(x_i, x)$$

A Counterexample

ex: If I plug in something that is symmetric, but it's not positive definite, into any algorithms that we want to kernelize, it would happen that the algorithm will not converge OR WORSE

↳ it will converge when it's POSITIVE AND it will not converge other times when it's not
⇒ BUG which will emerge IN A MOMENT of maximum disaster (?)

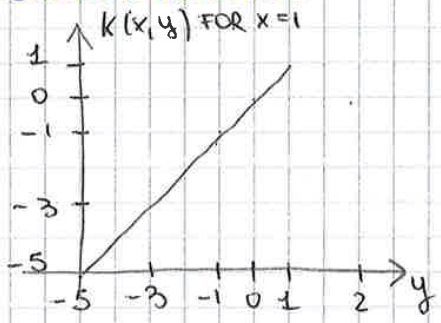
FOR instance → If we have a given function K which is symmetric

↓

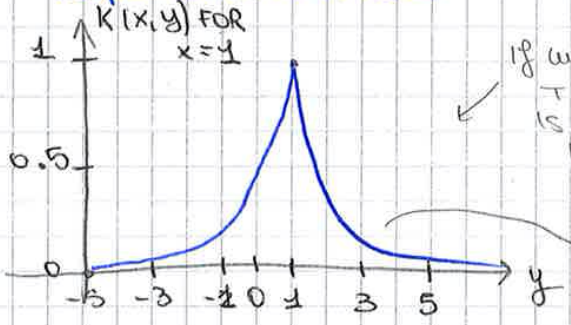
- **Simple Trick for checking Mercer's condition**
  Compute the Fourier transform of the Kernel and check that IT IS NOT negative

We can always compute the Kernel matrix
↳ we TEST IT ON a SUBSET, IN CERTAIN POINTS !!

Linear Kernel



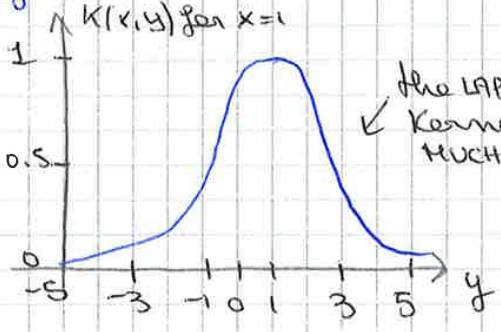
Laplacian Kernel



If we WANT THAT everything IS IN A certain RANGE AND WE WANT TO KEEP the TAILS

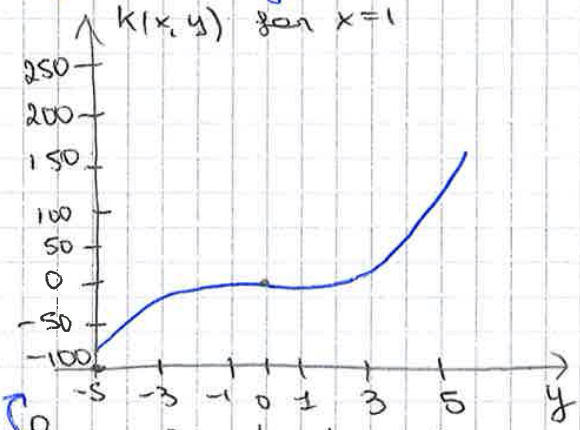
IF WE WANT OUR REPRESENTATION TO BE very COMPACT

Gaussian Kernel



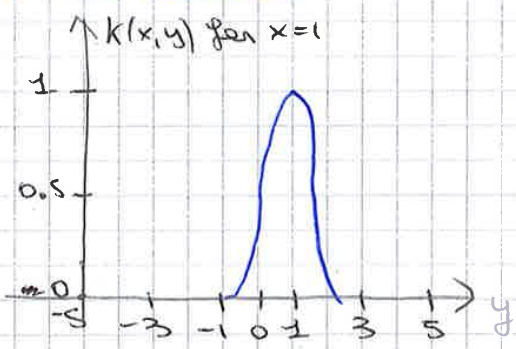
the LAPLACIAN Kernel IS MUCH FASTER

Polynomial of order 3



Remember that we have 3 parameters to set
(it's quite never used)
↳ IN computer VISION

B_3 SPLINE Kernel



Parzen Windows

DENSITY ESTIMATION

Even if our examples are very concrete (e.g. face recognition in PCA exercitation), we are always dealing WITH DATA that are RANDOM VARIABLES

↳ ex: PROBABILITY DISTRIBUTION which makes one the images of DOGS....

- Observe some DATA x_i
- WANT TO ESTIMATE $p(x)$ → PROBABILITY DISTRIBUTION

This is an example of what is called:

Curse of dimensionality (bis)

↳ When I have DISCRETE RANDOM VARIABLE, e.g.:
- ENGLISH, CHINESE, GERMAN, FRENCH, ...
- MALE, FEMALE
- ZIP CODE
- DAY of the week
- OPERATING SYSTEM
- ....

If I have a lot of variables and I WANT TO estimate the probabilities by the frequency, which is actually a good idea, the more things I WANT TO figure out → I WANT TO know people nationalities, where they live, what kind of OPERATING system they use.....

The more variables I WANT TO KNOW → I need more and more DATA
↳ more samples!

※ bins GROWS exponentially

This is not a problem related only to the discrete DOMAIN but also to the CONTINUOUS DOMAIN

CONTINUOUS RANDOM VARIABLES
- → INCOME
- → BANDWIDTH
- → TIME

→ need many bins per dimension

Density Estimation

Actually, the continuous domain seems to require an infinite number of bins → this is not the case, however

Curse of DIMENSIONALITY ↓
the higher is the dimension where the DATA live, the more are the number of DATA that I need
↳ in order to be able to estimate the probability distribution in a reliable way (modo affidabile)

** PCA → in our exercises we always assume that we have enough STATISTICS / enough DATA
Also in PCA we deal with RANDOM variables!
↳ we must have enough DATA!!

* If I have given training DATA, the moment that I start to compute something from these training data, I say: "I trust that I have enough data" → I have enough knowledges about the problem

* If your method is a DISCRIMINATIVE METHOD, which is what we did with the Perceptron meaning...
I am interested in the MAXIMUM likelihood
I am not interested on the POSTERIORS
I am interested in finding a separating hyperplane
Methods DATA DRIVEN
Then, even if the statistic is not strictly sufficient, probably it gonna be OK (?)

* If You are going to a GENERIC GENERAL APPROACH ⇒ we WANT to estimate the POSTERIORS →

↳ a different vector ($\pm\epsilon$) ⇒ zero

▶ This breaks if we see slightly different instances

So ⇒ ▶ Kernel density estimate

↳ it's something like this:

$$\int_X K_x(x')\,dx' = 1 \quad \forall x$$

Smear out empirical density with a nonnegative smoothing Kernel $K_x(x')$ SATISFYING:

$$\int_X K_x(x')\,dx' = 1 \quad \forall x$$

→ we substitute the δ with K AND that's ALL
→ this solution allows for some 'noise'

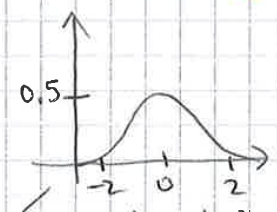
Density estimate:

$$p_{emp}(x) = \frac{1}{m}\sum_{i=1}^{m}\delta_{x_i}(x) \longrightarrow \hat{p}(x) = \frac{1}{m}\sum_{i=1}^{m}K_{x_i}(x)$$

↙ PARZEN WINDOWS FOR density estimate

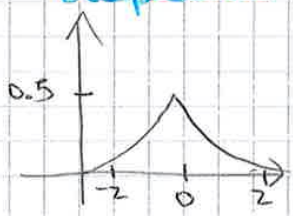
- Smoothing Kernels

GAUSS



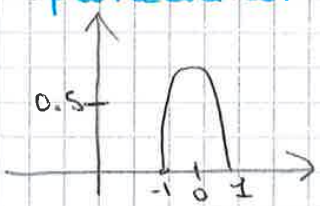
$(2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2}$

Laplace



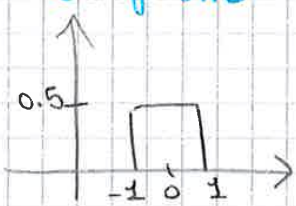
$\frac{1}{2} e^{-|x|}$

Epanechikov



$\frac{3}{4}\max(0, 1-x^2)$

Uniform



$\frac{1}{2}\chi_{[-1,1]}(x)$

example:
we put on each bin we have a gaussian
→ with a linear combination of gaussians this gives an estimate of probability

Nearest Neighbor ← Very useful!

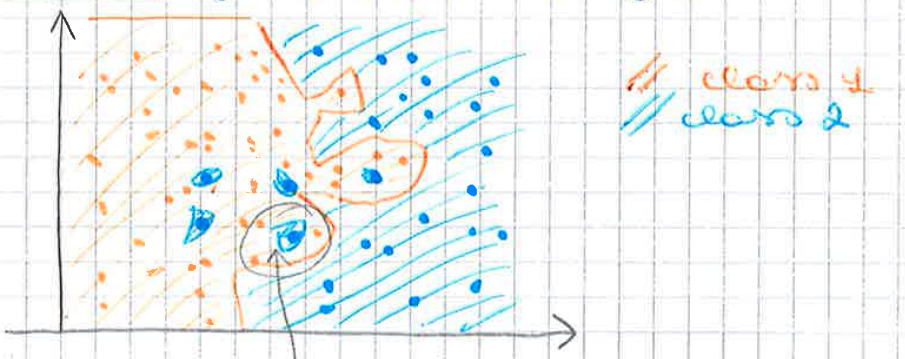
We are a lot of DATA
⇒ • TABLE LOOKUP → we build a table with ALL our training DATA (for instance: all the images of DOG...)
FOR PAIEVIOUSLY seen instance remember label
↳ see the images of all the closers that are we given

Then ⇒ • NEAREST neighbor
- PICK LABEL OF MOST SIMILAR NEIGHBOR
- SLIGHT IMPROVEMENT - USE K-NEAREST NEIGHBORS

THEN, When one tests the imagenet, you compute the distance between your TEST IMAGE AND ALL the IMAGES that are STORED IN class 1

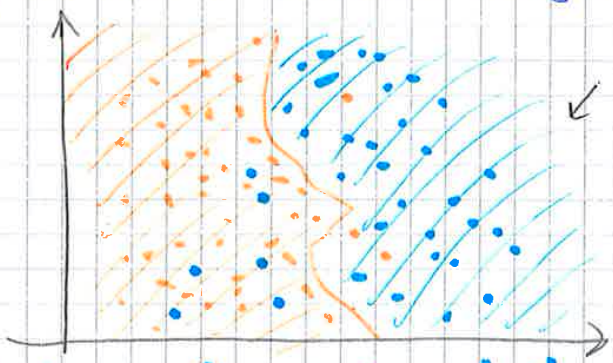
- Really useful heuristic!
- Easy to implement for small amount of data

Example of: 1-Nearest Neighbor



I could have a mistake in these areas! → That's a challenge!

If we use: 4-Nearest Neighbor

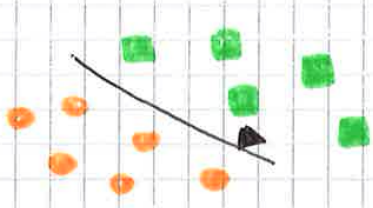


Actually we are making mistakes
↳ We have outliers (?)

If we get more data:

There are some proofs about the convergence of this algorithm, but we won't do in details

with 1 Nearest and 4 Nearest Neighbors



1 nearest Neighbor

- converges to perfect solution if separation
- Twice the minimal error rate $2p(1-p)$ for noisy problems

K-Nearest Neighbor

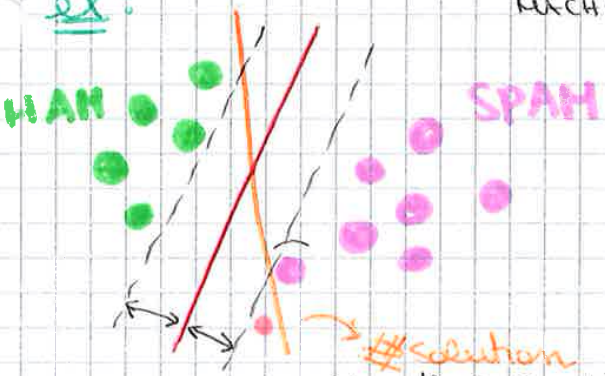
- converges to perfect solution if separation (BUT needs more DATA)
- converges to minimal error: $\min(p, 1-p)$ for noisy problems. (use increasing K)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 244

AND IF new data arrive, of course I can have bad classification but the key expectation is that this solution will work better than the others ⇒ THIS IS the INTUITION AND SUPPORT VECTOR ~~ELECTRO~~ MACHINES

↳ because this IS NOT a perfect classifier

ex:



solution

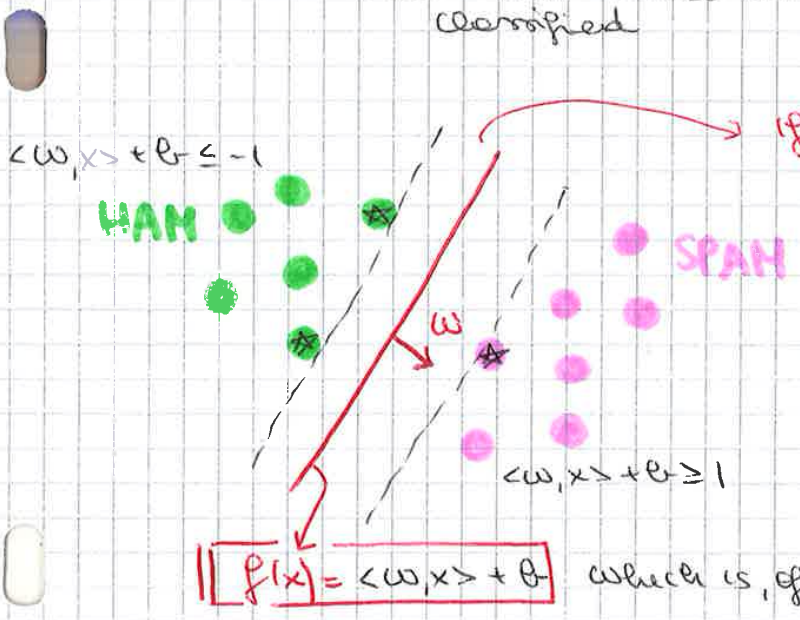
ALSO this IS A LEGITIMATED solution but PROBABLY if A POINT ● ← LIKE THIS arrives from one distribution ⇒ this POINT could badly classified

Support Vector Machines

among all the permitted linear separators I WANT to choose the one that gives the LARGEST MARGIN among the 2 DISTRIBUTIONS that we WANT to separate

⇓

we WANT TO FIND the OPTIMAL LINEAR SEPARATOR



if this line → is defined by a vector w

I expect that on the side of HAM, I would have: $\langle w, x \rangle + b \leq -1$

AND on the side of SPAM: $\langle w, x \rangle + b \geq 1$

$f(x) = \langle w, x \rangle + b$ which is, of course, a linear function

What ~~DOES~~ makes different this solution/situation with respect to what we've seen so far?

What we are DOING different here is that we are explicitely finding some POINTS in the 2 classes for which we're saying "well, something there's ese different from the others, they are not like the others" (POINTS: ⊛)

↳ These POINTS are the ones which define the lines: - - ------

And, as a consequence, they define the solution: ———

The Red line is the line which is exactly in the middle between these: - - - - - -

↳ it has this

↳ lines that have the minimum distance from the closest POINTS of the 2 DISTRIBUTIONS

OPTIMIZATION PROBLEM: (we are going to solve)

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to:} \quad y_i[\langle x_i, w\rangle + b] \geq 1$$

Dual Problem

THIS OPTIMIZATION PROBLEM is what is called:

- **PRIMAL OPTIMIZATION PROBLEM**

We need to minimize:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to:} \quad y_i[\langle x_i, w\rangle + b] \geq 1$$

To solve this, we're going to use: **Lagrange Multipliers**

Lagrange Multipliers → it's a strategy to find the local maxima and minima ~~just of~~ of a function that is subject to equality constraints

⇓

We have the function: $f(x,y)$ and we have to maximize it
We have also a constraint:

$g(x,y)$ is subjected to $g(x,y)=0$

Assuming that these functions (f AND g) have continuous partial derivatives, then:

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$

(When we find the points that satisfy this equation, we have found the POINTS ~~of the optimization~~ that SATISFY the constraint) (?)

$\mathcal{L}$ → Lagrangian function

So, let's write the Lagrangian: (it defines the DUAL PROBLEM)

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i[\langle x_i, w\rangle + b] - 1]$$

↳ this is my constraint (I have i constraints)

Now, I need to FIND the derivatives of $\mathcal{L}$ with respect to w and b, I have to put them to zero
And I have to FIND the POINTS in α_i that would give the optimality in w and in b:

- LAGRANGIAN FUNCTION: $\mathcal{L}(w,b,\alpha)$
- → OPTIMALITY IN w,b is at saddle POINT with α
- Derivatives in w,b need to vanish

▶ So let's BEGIN doing the Derivatives:

$$\frac{\partial}{\partial w}\mathcal{L}(w,b,\alpha) = \frac{1}{2}\cdot 2\|w\| - \frac{\partial}{\partial w}\left[\sum_i \alpha_i y_i \langle x_i, w\rangle + \sum_i \alpha_i y_i b - \sum_i \alpha_i\right]$$

this term depends on w linearly

they don't depend on w

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 244

Is this linear combination going to be among ALL my training DATA?
THIS IS what we DO with A PERCEPTRON
We use all the training ~~the~~ DATA for the solution with a perceptron
↳ But, this is not what we've found:

We have to maximize:

$$\max: -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0,\ \alpha_i \geq 0$$

When I solve this optimization problem, I will find that most of values of α are zero
→ the only α_i different from zero are the ones that multiply the ~~yellow~~ ORANGE VECTORS (●)
↳ which are the vectors lieing on the MARGIN

So, the solution w will be a linear combination of only the points of the 2 classes that lie on the MARGINS
These vectors are the vectors that support my decision function AND this is the reason why my classifier is called: SUPPORT VECTOR MACHINE

This classifier, actually, compresses the DATA that I have Because it will use only a SUBSET of the POINTS AND it might be a very small subset

⇒ The solution we find in this way, it will be OPTIMAL in terms of minimizing the RISK

PREVIEW: When I say that I minimize the RISK, this is the empirical RISK which is the RISK we've already defined → the RISK of making mistakes on the training DATA
But I also minimize the Structural RISK which is the RISK of making mistakes of any possible future data
→ We will see that is the BEST APPROXIMATION that we can give at the Bayes RISK which is the MINIMUM RISK, the BEST RISK that we can estimate with a classifier (IN the LIMIT for number of TRAINING DATA → ∞, this STRUCTURE RISK approximates the Bayes RISK AND this is why we know that we are minimizing the STRUCTURAL RISK)
↳ with this classifier

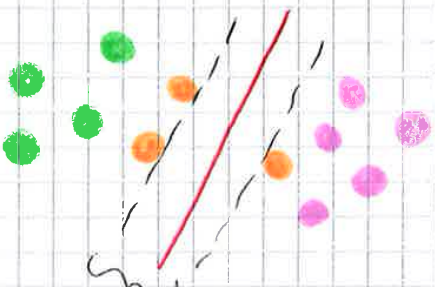
This is why that, for a very very very long time, support vector machines was the state of the art classifier in research and commercial APPLICATIONS
The only reason why support vector machine is not used today AND Deep learning is preferred is that to make scalable non linear SUPPORT VECTOR MACHINE → it DOESN'T SCALE for ~~for~~ over milions and milions of training DATA
↳ This is the ONLY REASON!
If one day we will be able to overcom this problem, support vector machine will be preferred because it provides the ~~garanto~~ theoretical guarantees (Deep learning NOT)

If we have not the theoretical guarantees it means that we don't know what IS GOING TO DO (ex: we TAKE A TAXI BUT WE DON'T KNOW IF THE DRIVER HAS THE DRIVE LICENCE OR NOT)

example: let's suppose that we have some DATA (but they aren't enough to describe the STATISTICS) → a technique also used in DEEP LEARNING is called:

Remark: DATA AUGMENTATION

So, I have some data AND I apply some transformations (e.g.: I can ROTATE the image, I can change the scale, I can add some noise, I can change the illumination ....) → IN SVM I know that all that matters are the SUPPORT VECTORS, SO I could think to eliminate all only the SUPPORT VECTORS



IN general if I am expecting to have a problem ⇒ this PROBLEM IS GOING TO BE here

So, I ~~perturb~~ my DATA
↳ I "move" the DATA
⇒ I PERTURB my SOLUTION
(to find where the problems are going to be)
↳ TO FIND A better solution
to do this, the best way is DOING alike this

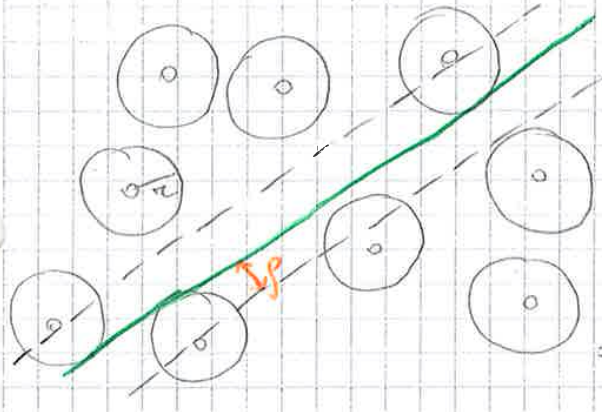
Remark: in the example → I need only the 3 POINTS on the MARGINS BUT I really need 3 POINTS to define a line? (let's suppose that our space is actually a plane AND our vectors are POINTS) ⇒ NO, I need only 2 POINTS to define a line
⇒ further compression of the DATA!

So, if I find that the number of support vectors ~~are~~ IS higher than the dimension of my solution ⇒ I can further reduce my vectors → I can further compress

Remark: If, on the contrary, I have memory LIMITATIONS to store all the real SUPPORT vectors that actually I need → I can reduce the number of these AND do an APPROXIMATION ⇒ IT WORKS quite & well (something as PCA)

These support vectors give a lot of tools to improve the performance AND to make the things simpler

SO, WHY LARGE MARGINS?

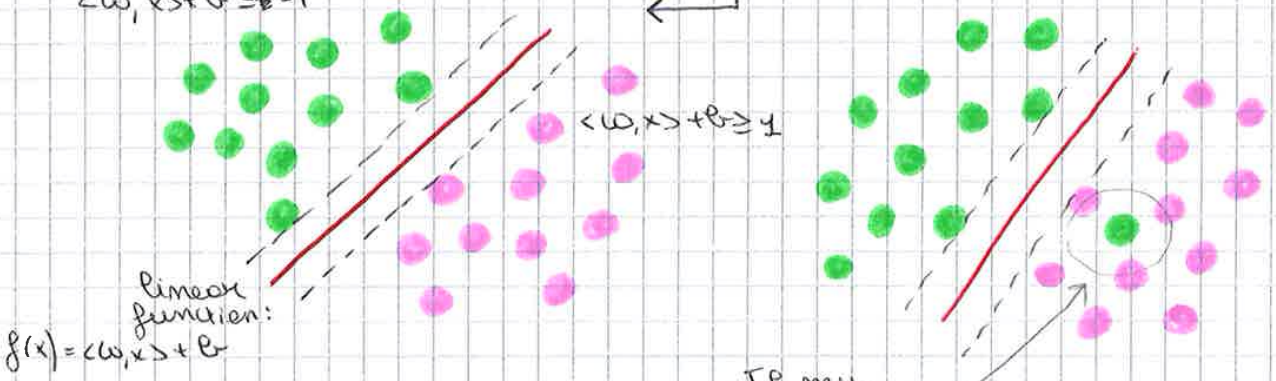


If I look at each given POINT I could have a more nearest neighbours approach
↳ BASICALLY, I look at the POINTS AND I define a RADIUS AND AS LONG AS the distance between ~~the~~ A query POINT and one of the POINT is below r, I know that is closer to that POINT than any other

⇒ Why do we WANT TO introduce large MARGINS with respect to nearest neighbours?

Soft Margin Classifiers

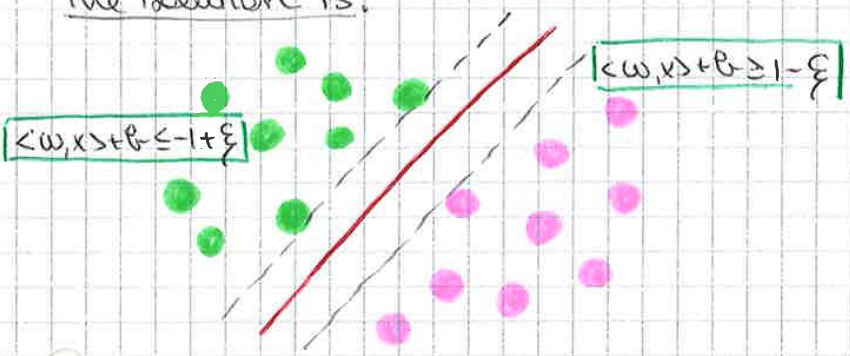
This is what we've seen so far: Large Margin CLASSIFIER $\langle w, x\rangle + b \leq -1$ ⇒ NOW:



If my training DATA include this POINT
⇒ linear separator is impossible

So, what DO we do?
How to manage this type of problem?
(Actually it can be kernelized)

The solution is:



It is like to consider the TRAINING DATA as real DATA with some noise (for instance) → so, it's like to allow us to make some mistakes
→ I make the margins SOFT

When I do that, I do a CONVEX OPTIMIZATION PROBLEM
Actually, BY ADDING A variable ξ we make the margin SOFT (≈ we are GOING to ALLOW some mistakes)
We can solve these problem exactly with the same technique that we've seen so far

What we will need is → to: MINIMIZE the AMOUNT of SLACK

ADDING SLACK VARIABLES

* Our hard margin problem was:

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i[\langle w, x_i\rangle + b] \geq 1$$

NOW → with SLACK VARIABLES, the PROBLEM becomes:

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \quad \text{subject to:} \quad y_i[\langle w, x_i\rangle + b] \geq 1 - \xi_i, \ \xi_i \geq 0$$

The PROBLEM is always feasible