# ML with numerical optimization
# brief introduction

Gilles Richard

IRIT-CNRS, Toulouse University, France

# What is this ?

example 1:


| sépal | | pétal | | |
|---|---|---|---|---|
| long. | larg. | long. | larg. | type |
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| . | . | . | . | . |
| 7.0 | 3.2 | 4.7 | 1.4 | versic. |
| 6.4 | 3.2 | 4.5 | 1.5 | versic. |
| . | . | . | . | . |
| 6.3 | 3.3 | 6.0 | 2.5 | virgin. |
| 5.8 | 2.7 | 5.1 | 1.9 | virgin. |
| . | . | . | . | . |

Aim: find a simple numerical function $f$ such that:

- computing $f(x)$ provides the class of $x$
- $x$ is a vector of numerical values (i.e. a row in the table)
- the class is just 0 or 1 (or 1 and -1)
- for instance:

$$f(x) = \alpha sepal.long + \beta speal.larg + \dots$$

- Classifier:

# Another example: letters and pixels

Example 2:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 39 | 39 | 39 | 38 | 37 | 37 | 36 | 36 | 35 | 34 | 34 | 34 | 33 | 33 | 33 |
| 32 | 31 | 31 | 30 | 30 | 29 | 28 | 28 | 27 | 26 | 27 | 26 | 26 | 25 | 25 | 25 |
| 24 | 23 | 23 | 22 | 21 | 19 | 18 | 16 | 15 | 17 | 20 | 21 | 22 | 23 | 18 | 14 |
| 25 | 28 | 25 | 23 | 23 | 22 | 21 | 19 | 18 | 16 | 13 | 13 | 13 | 12 | 17 | 19 |
| 18 | 19 | 14 | 18 | 21 | 22 | 23 | 24 | 25 | 25 | 25 | 25 | 26 | 26 | 26 | 27 |
| 28 | 28 | 28 | 28 | 29 | 30 | 30 | 30 | 32 | 32 | 32 | 32 | 32 | 31 | 33 | 33 |
| 34 | 35 | 36 | 36 | 36 | 37 | 37 | 37 | 38 | 39 | 39 | 39 | 39 | 39 | 41 | 40 |
| 40 | 41 | 41 | 41 | 40 | 42 | 41 | 41 | 41 | 41 | 42 | 41 | 42 | 42 | 41 | 41 |
| 40 | 40 | 39 | 38 | 38 | 37 | 37 | 36 | 36 | 35 | 35 | 35 | 33 | 34 | 32 | 32 |
| 31 | 31 | 30 | 29 | 29 | 29 | 29 | 28 | 28 | 27 | 27 | 27 | 26 | 25 | 25 | 25 |
| 24 | 24 | 23 | 22 | 22 | 20 | 19 | 17 | 16 | 18 | 22 | 22 | 24 | 24 | 19 | 13 |
| 25 | 28 | 26 | 24 | 24 | 23 | 22 | 21 | 20 | 18 | 16 | 15 | 14 | 13 | 18 | 19 |

- a matrix of pixel, each pixel is a grey level (black and white image)
- each image is a letter
- find $c$ such that $f(x_1, \dots, x_n) \in \{1, -1\}$ tells us if the image is letter a: too hard
- find $h$ such that $h(x_1, \dots, x_n) \in \mathbb{R}$ and $\forall z = (x, y) \in TS, h(x) \times y > 0$

# How to compute $f$ ?

## What do we have?

- as usual, a finite set of examples $TS = \{z_1, \ldots, z_m\}$ with their class
- $class : \mathbb{R}^n \to \{1, -1\}$
- if no noise, $z = (x, class(x))$, if noise we can have $z = (x, y)$ with $y \neq class(x)$

# How to compute $f$ ?

## What do we have?

- as usual, a finite set of examples $TS = \{z_1, \ldots, z_m\}$ with their class
- $class : \mathbb{R}^n \rightarrow \{1, -1\}$
- if no noise, $z = (x, class(x))$, if noise we can have $z = (x, y)$ with $y \neq class(x)$

## Separable versus linearly separable

Just give an example with $\mathbb{R}^2$ and explain

separable: we can find a suitable $f$

linearly separable: among all the suitable $f$, there is an hyperplan (a line in $\mathbb{R}^2$)

Give a circle and a line as examples

Example of a rectangle as separator (i.e. the normal patients for medical data)

# A first simple algorithm (perceptron)

**to start...**

- Look for $h(a, b)(x) = a.x + b = a_1 x_1 + a_2 x_2 + ::: + a_n x_n + b$
- init: $a \leftarrow (0, \ldots, 0)$; $b \leftarrow 0$; done $\leftarrow$ false;
  while not done
  done $\leftarrow$ true;
  for each $(x, y) \in TS$, if $(a.x + b).y \leq 0$ //disagreement
  then
  $a \leftarrow a + \tau.y.x$
  $b \leftarrow b + \tau.y$; done $\leftarrow$ false
  return $(a, b)$
  $\tau > 0$ (convergence speed tuning)

# A first simple algorithm (perceptron)

## to start...

- Look for $h(a,b)(x) = a.x + b = a_1 x_1 + a_2 x_2 + ::: + a_n x_n + b$
- init: $a \leftarrow (0, \ldots, 0)$; $b \leftarrow 0$; done $\leftarrow$ false;
  while not done
  done $\leftarrow$ true;
  for each $(x, y) \in TS$, if $(a.x + b).y \leq 0$ //disagreement
  then
  $a \leftarrow a + \tau.y.x$
  $b \leftarrow b + \tau.y$; done $\leftarrow$ false
  return $(a, b)$
  $\tau > 0$ (convergence speed tuning)

## properties

1. simple...
2. incremental
3. convergence iff linearly separable

# Using Gradient Descent: a better algorithm

## to start again

- Again $h(a, b)(x) = a.x + b = a_1x_1 + a_2x_2 + ::: + a_nx_n + b$
- Minimize empirical error on $TS$: $e_{emp} = \frac{1}{n}\Sigma|(h(a, b)(x) - y)|$
- Simpler to minimize on $TS$,
  $E(a, b)(TS) = \frac{1}{2}\Sigma_{(x,y)\in TS}(h(a, b)(x) - y)^2$
- init: $a \leftarrow (0, \ldots, 0)$; $b \leftarrow 0$; $\epsilon$ (precision) $\tau$ (step size)
  while $\Delta e_{emp} > \epsilon$ (or $\Delta E > \epsilon$)
  for each $(x, y) \in TS$,
  $a \leftarrow a + \tau.(y - (a.x + b)).x$ $(i.e. \frac{\delta E(a,b)(TS)}{\delta a})$
  $b \leftarrow b + \tau.(y - (a.x + b))$ $(i.e. \frac{\delta E(a,b)(x)}{\delta b})$;
  end while return $(a, b)$

# Using Gradient Descent: a better algorithm

## to start again

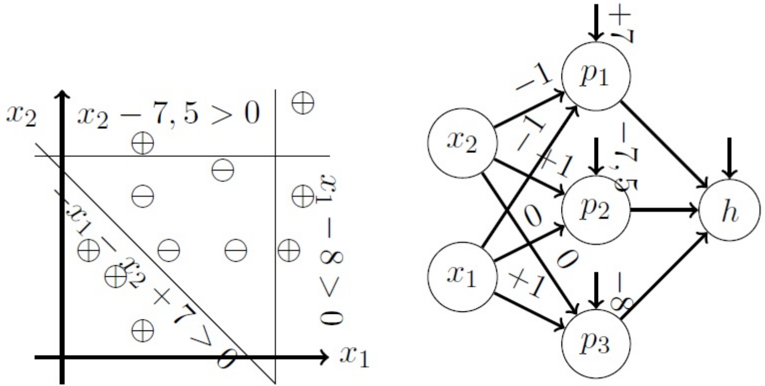
- Again $h(a,b)(x) = a.x + b = a_1x_1 + a_2x_2 + ::: + a_nx_n + b$
- Minimize empirical error on $TS$: $e_{emp} = \frac{1}{n}\Sigma|(h(a,b)(x) - y)|$
- Simpler to minimize on $TS$,
  $E(a,b)(TS) = \frac{1}{2}\Sigma_{(x,y)\in TS}(h(a,b)(x) - y)^2$
- init: $a \leftarrow (0,\ldots,0); b \leftarrow 0; \epsilon$ (precision) $\tau$ (step size)
  while $\Delta e_{emp} > \epsilon$ (or $\Delta E > \epsilon$)
  for each $(x,y) \in TS$,
  $a \leftarrow a + \tau.(y - (a.x + b)).x\ (i.e.\frac{\delta E(a,b)(TS)}{\delta a})$
  $b \leftarrow b + \tau.(y - (a.x + b))\ (i.e.\frac{\delta E(a,b)(x)}{\delta b})$;
  end while return $(a,b)$

## properties

1. convergence whatever the situation
2. convergence can be very very slow !
3. other minimization functions (log, etc.)

# Neural Networks howto

## The big idea...



# Neural Networks ... propagate

## One node per function

- a finite set $\mathcal{N}$ of nodes
- $n$ variables then $n$ input nodes: they do not classify - 1 output node $h$
- Each internal node $m$ provides a classifier
- Combining the classifiers provides the resulting hypothesis $h$
- $\forall l \in \mathcal{N}$, (except input nodes - recursive definition)

$$l(x) = sign(\Sigma \alpha_{ml} m(x) + \beta_l)$$

  where $\beta_l$ is the weight of node $l$ and $\alpha_{ml}$ the weight of the edge from node $m$ to node $l$ (i.e. $m$ is a parent of $l$)
- Propagate the answer (the class) from input to output $h$
- Retro-propagate the error $y - h(x)$ from output $h$ to input

# Neural Networks ... retro-propagate

## Learning the weight !

- for all $(x, y) \in TS$
- for all $l \in \mathcal{N}$, compute $l(x)$ (then $h(x)$ is computed)
- $\delta h \leftarrow \tau(y - h(x))$
- //retro-propagate the final error
  For each hidden node $l$, (recursive definition)
  $\delta l = \Sigma \alpha_{lm} \delta m$ for $m$ child of $l$ where $\beta_l$ is the weight of node $l$ and $\alpha_{ml}$ the weight of the edge from node $m$ to node $l$
- //update the weight coefficients
  $\forall l \in \mathcal{N}, \alpha l \leftarrow \alpha_l + \delta_l$
  $\beta_l \leftarrow \beta_l + \delta_l$
-

# Neural Networks pro/cons

## Pro

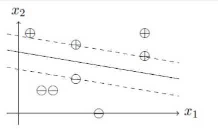
- Very successfull for diverse tasks
- Can use other "neuronal" functions (i.e. non linear classifiers)
  Ex.: sigmoid $sigm(a) = \frac{1}{1+e^{-\lambda a}}$ and replace *sign* with *sigm*
- On the shelves product (weka for instance!)
- Revival with "deep learning"

## Cons

- Empirical init of the nodes and structure (done by human experts...)
- Empirical init of the weights and very long training time
- Tuning very difficult
- Halting condition

# Support Vector Machines (V. Vapnik - 1979, etc.)

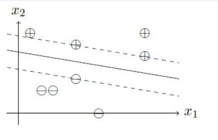
## Another clever idea !




If linearly separable, there is a kind of "separating PIPE"

# Support Vector Machines (V. Vapnik - 1979, etc.)

## Another clever idea !




If linearly separable, there is a kind of "separating PIPE"

## Let us think about it

- Infinite number of separating hyperplans
- The "best" one could be the one maximizing the margin !
- Support vectors = the examples on the pipe limits!
- We don't care about the other examples...

# SVM: some facts

## The problem...

- Notion of margin associated to a separating hyperplan $H$
- Ultimately find the hyperplan with the biggest margin...
- Hyperplan $H$ with equation $a.x + b = 0$ with normalized normal vector $w = \frac{a}{||a||}$
- Given $z \in TS$, distance $d(z, H)$ to be computed
- Let $y \in H$ such that $d(z, H) = d(z, y)$ i.e. $y = z - d(z, H).\frac{a}{||a||}$
- Then $a.y + b = 0$ i.e. $a.(z - d(z, H).\frac{a}{||a||}) + b = 0$ i.e. $d(z, H) = \frac{a.z+b}{||a||}$
- geometric margin for $z$ : $marg(z, H) = cl(z).(\frac{a.z+b}{||a||})$
  geometric margin for $H$ : $min\{marg(z, H)|z \in TS\}$
- Maximizing $marg(H)$ equivalent to maximize $\frac{1}{||a||}$ i.e. minimize $\frac{1}{2}||a||^2$

# SVM: the algorithm

## The problem...

- Initial pb.: unknown $a$ and $b$ ($n+1$ variables) - then minimize $||a||^2$
- Constraints: $\forall(x,y) \in TS, y \times (a.x+b) \geq 1$
- Classical math exercise... can be solve in $\mathcal{O}(n^3)$ and unique solution
- On the shelve software to do the job !
- Convergence if linearly separable... STUNNING PERFORMANCES...
- and ... IF NOT ?????????????????????????

# SVM for non linerarly separable data: how to ?

### The problem...

- $TS = \{1, +1), (2, +1), (4, -1), (5, -1), (8, +1), (9, +1)\}$
- Projection in $\mathbb{R}$ with $\phi(x) = (x, x^2)$... linearly separable
- The main idea...

### New optimization problem to be solved!

- Find $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that
  - $\exists (a, b) \in \mathbb{R}^m \times \mathbb{R}$ with
  - $\forall (x, y) \in TS, y.(a.\phi(x) + b) \geq 1$
- But only $K(x, x') = \phi(x).\phi(x')$ needed (. is the scalar product also denoted $\phi(x)^T \phi(x')$)
- $K$ kernel function (kernel trick)

# SVM on the shelves

## Usual kernels

- Gaussian kernel $K(x, y) = \frac{e^{-||x-y||^2}}{2\sigma^2}$
- Polynomial kernel $K(x, y) = (x.y + 1)^p$
- Combination of kernels $K + K', K \times K', exp(K(x, x')), \ldots$
- "Magic" extension of SVM...

## Applications

- Weka has SVM included (LibSVM weka)
- $K$ kernel function representing "similarity" between examples...
- Example outside $\mathbb{R}^n$: $A$ a subset of a finite universe $\mathcal{A} : K(A, B) = 2^{|A \cap A'|}$

# Concluding remarks

- A lot of other options and methods (SMO, Ridge regression, etc.)
- Generally black box approach
- Parameter tuning: highly difficult and empirical task
- A lot of commercial implementations as well