Perceptron com *kernel*

Gaspar J. Machado

Departamento de Matemática, Universidade do Minho

março de 2024

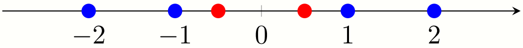
Motivação

- O algoritmo do modelo linear *Perceptron*, em todas as versões apresentadas, converge para um hiperplano que classifica bem todas as ocorrências de uma base de dados se e só se a base de dados é linearmente separável.
- As bases de dados reais usualmente não são linearmente separáveis.
- "Qual a relevância do algoritmo *Perceptron* se nem a tão simples base de dados associada ao operador lógico XOR consegue resolver?", perguntava-se nos anos 60 do século passado, período conhecido pelo *AI winter* também por este motivo (*cf.* Minsky, Marvin; Papert, Seymour (1969). "*Perceptrons: An Introduction to Computational Geometry*". The MIT Press).
- Apenas em meados dos anos 80 do século passado apareceram soluções para ultrapassar o problema da não separabilidade linear dos dados, nomeadamente:
  - *Perceptron* multicamada (*multilayer Perceptron* — MLP) (que é um nome enganador).
  - linearização dos dados aumentando a dimensão do espaço dos atributos.

Ex7 $(I = 1)$ | D

- Base de dados: $D = (x^n, y^n)_{n=1}^6$ com

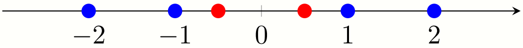
$$x^1 = -2 \qquad y^1 = -1$$
$$x^2 = -1 \qquad y^2 = -1$$
$$x^3 = -\tfrac{1}{2} \qquad y^3 = +1$$
$$x^4 = \tfrac{1}{2} \qquad y^4 = +1$$
$$x^5 = 1 \qquad y^5 = -1$$
$$x^6 = 2 \qquad y^6 = -1$$



Ex7 $(I = 1)$ | D

- Base de dados: $D = (x^n, y^n)_{n=1}^{6}$ com

$$
\begin{aligned}
x^1 &= -2 & \quad y^1 &= -1 \\
x^2 &= -1 & \quad y^2 &= -1 \\
x^3 &= -\tfrac{1}{2} & \quad y^3 &= +1 \\
x^4 &= \tfrac{1}{2} & \quad y^4 &= +1 \\
x^5 &= 1 & \quad y^5 &= -1 \\
x^6 &= 2 & \quad y^6 &= -1
\end{aligned}
$$



- D não é linearmente separável.

Ex7 $(I = 1)$ | linearização | D'

- Nova base de dados: $D' = (z^n, y^n)_{n=1}^{6}$ com

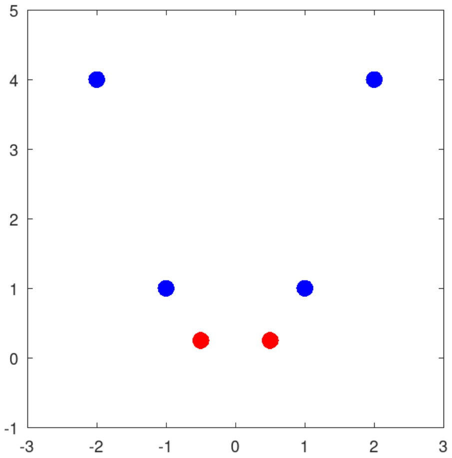
$$z^n = \Phi(x^n) = (x^n, (x^n)^2)^\top,$$

ou seja,

$$\begin{array}{ll}
z^1 = \Phi(x^1) = \Phi(-2) = (-2, 4)^\top & \quad y^1 = -1 \\
z^2 = \Phi(x^2) = \Phi(-1) = (-1, 1)^\top & \quad y^2 = -1 \\
z^3 = \Phi(x^3) = \Phi(-\frac{1}{2}) = (-\frac{1}{2}, \frac{1}{4})^\top & \quad y^3 = +1 \\
z^4 = \Phi(x^4) = \Phi(\frac{1}{2}) = (\frac{1}{2}, \frac{1}{4})^\top & \quad y^4 = +1 \\
z^5 = \Phi(x^5) = \Phi(1) = (1, 1)^\top & \quad y^5 = -1 \\
z^6 = \Phi(x^6) = \Phi(2) = (2, 4)^\top & \quad y^6 = -1
\end{array}$$

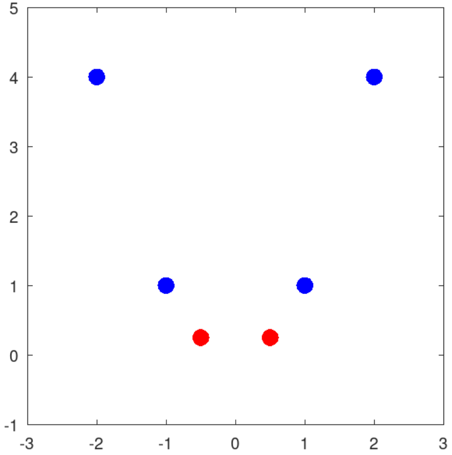
Ex7 $(I = 1)$ | D'

- Nova base de dados: $D' = (z^n, y^n)_{n=1}^{6}$, $z^n = \Phi(x^n) = (x^n, (x^n)^2)^\top$



Ex7 $(I = 1)$ | D'

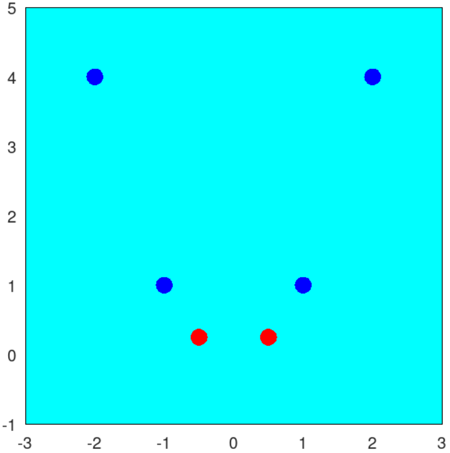
- Nova base de dados: $D' = (z^n, y^n)_{n=1}^{6}$, $z^n = \Phi(x^n) = (x^n, (x^n)^2)^\top$



- D' já é linearmente separável.

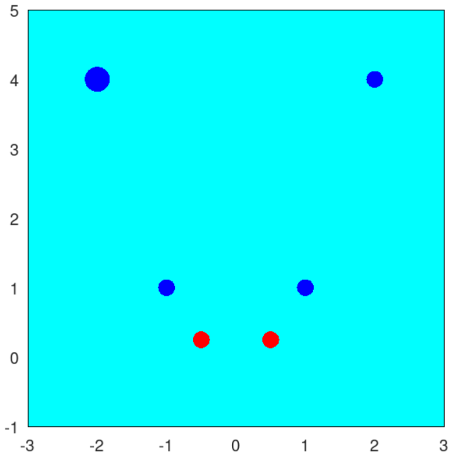
Ex7 $(I = 1)$ | Perc–v1 | $t = 0$

- $\tilde{w}_{(0)} = (0, 0, 0)^\top, E_{(0)} = 0.33333$



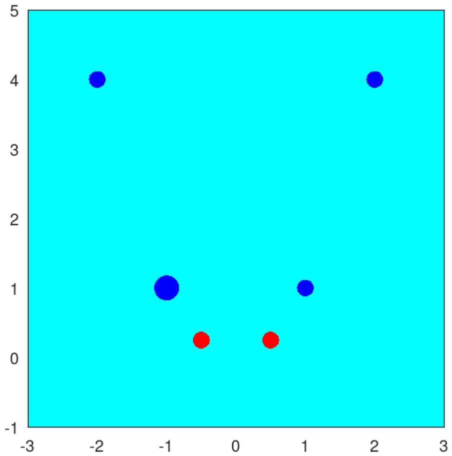
Ex7 $(I = 1)$ | Perc–v1 | $t = 1$

- $n = 1, \tilde{w}_{(1)} = (0, 0, 0)^\top, E_{(1)} = 0.33333$



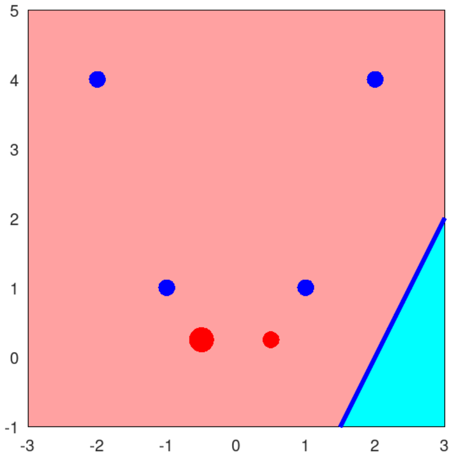
Ex7 $(I = 1)$ | Perc–v1 | $t = 2$

- $n = 2, \tilde{w}_{(2)} = (0, 0, 0)^\top, E_{(2)} = 0.33333$



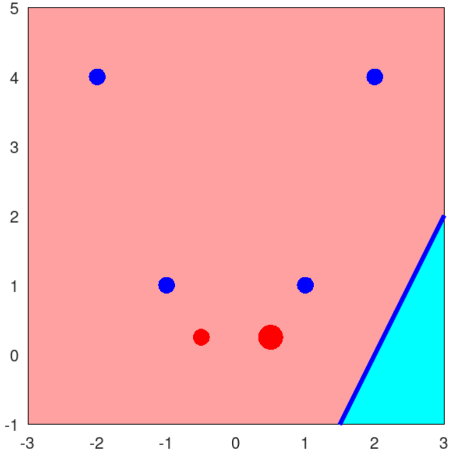
Ex7 $(I = 1)$ | Perc–v1 | $t = 3$

- $n = 3, \tilde{w}_{(3)} = (1, -0.5, 0.25)^\top, E_{(3)} = 0.66667$



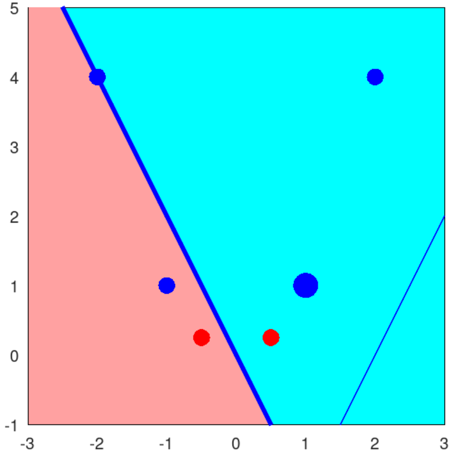
Ex7 $(I = 1)$ | Perc–v1 | $t = 4$

- $n = 4, \tilde{w}_{(4)} = (1, -0.5, 0.25)^\top, E_{(4)} = 0.66667$



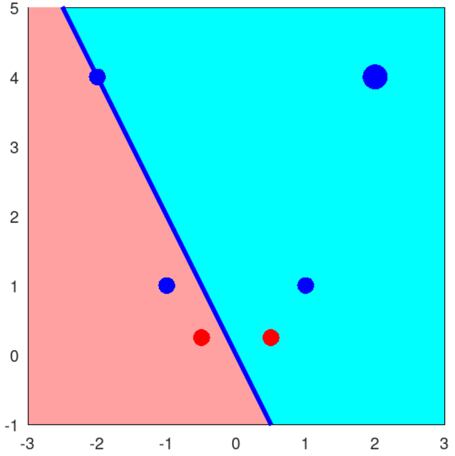
Ex7 $(I = 1)$ | Perc–v1 | $t = 5$

- $n = 5$, $\tilde{w}_{(5)} = (0, -1.5, -0.75)^\top$, $E_{(5)} = 0.33333$



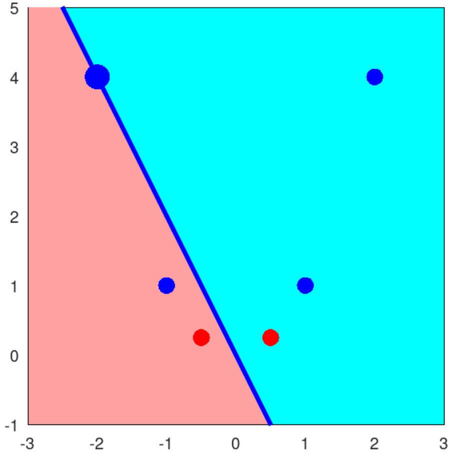
Ex7 $(I = 1)$ | Perc–v1 | $t = 6$

- $n = 6, \tilde{w}_{(6)} = (0, -1.5, -0.75)^\top, E_{(6)} = 0.33333$



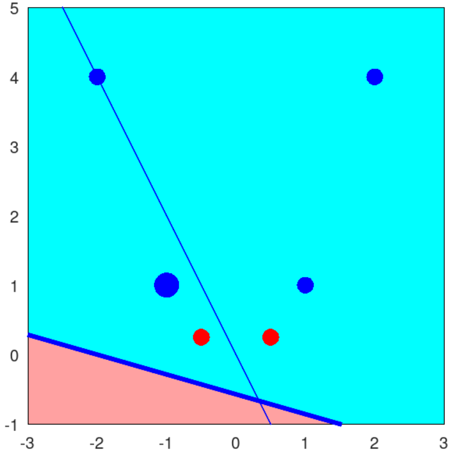
Ex7 $(I = 1)$ | Perc–v1 | $t = 7$

- $n = 1, \tilde{w}_{(7)} = (0, -1.5, -0.75)^\top, E_{(7)} = 0.33333$



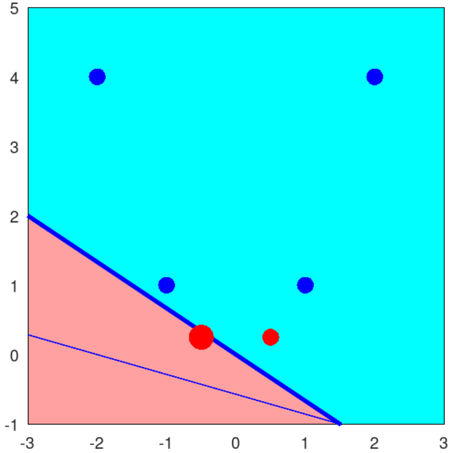
Ex7 $(I = 1)$ | Perc–v1 | $t = 8$

- $n = 2, \tilde{w}_{(8)} = (-1, -0.5, -1.75)^\top, E_{(8)} = 0.33333$



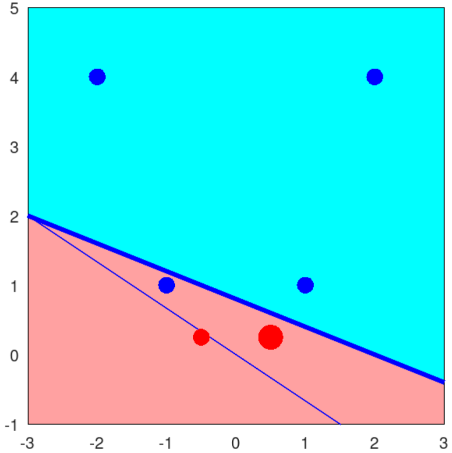
Ex7 $(I = 1)$ | Perc–v1 | $t = 9$

- $n = 3, \tilde{w}_{(9)} = (0, -1, -1.5)^\top, E_{(9)} = 0.16667$



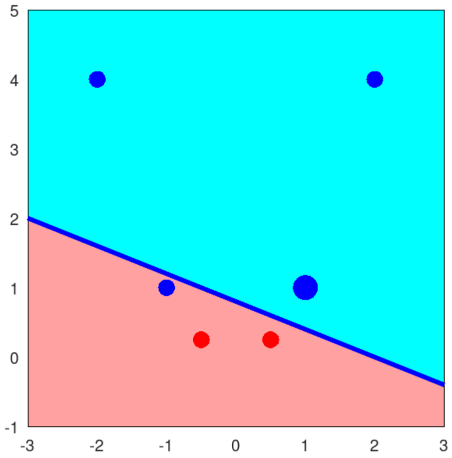
Ex7 $(I = 1)$ | Perc–v1 | $t = 10$

- $n = 4, \tilde{w}_{(10)} = (1, -0.5, -1.25)^\top, E_{(10)} = 0.16667$



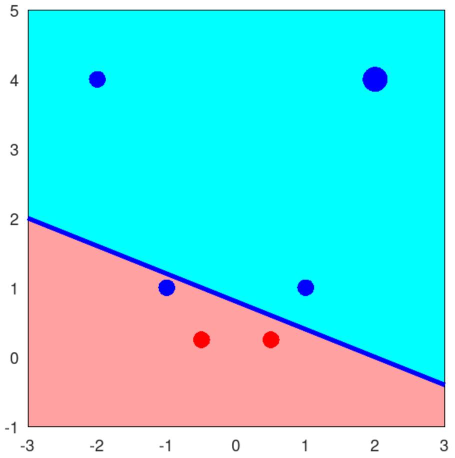
Ex7 ($I = 1$) | Perc–v1 | $t = 11$

- $n = 5, \tilde{w}_{(11)} = (1, -0.5, -1.25)^\top, E_{(11)} = 0.16667$

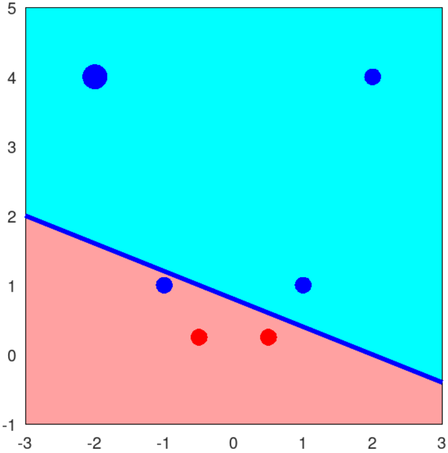


Ex7 $(I = 1)$ | Perc–v1 | $t = 12$

- $n = 6, \tilde{w}_{(12)} = (1, -0.5, -1.25)^\top, E_{(12)} = 0.16667$

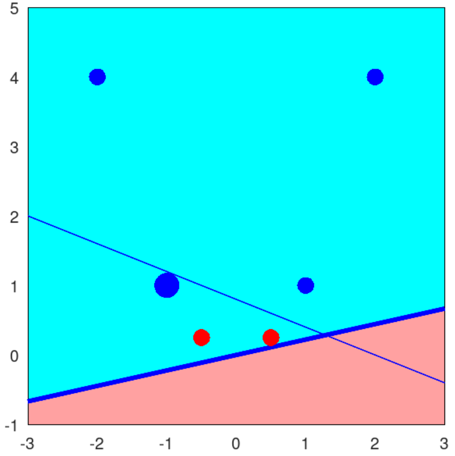


- $n = 1, \tilde{w}_{(13)} = (1, -0.5, -1.25)^\top, E_{(13)} = 0.16667$



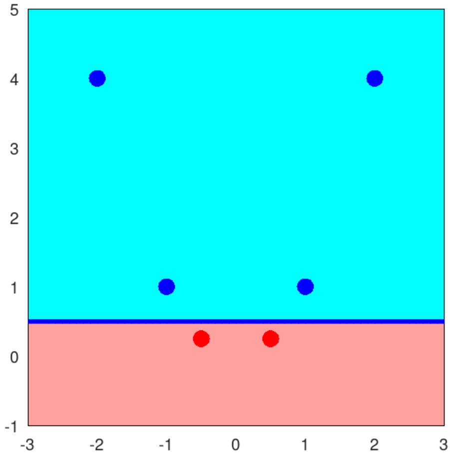
Ex7 $(I = 1)$ | Perc–v1 | $t = 14$

- $n = 2, \tilde{w}_{(14)} = (0, 0.5, -2.25)^\top, E_{(14)} = 0.33333$



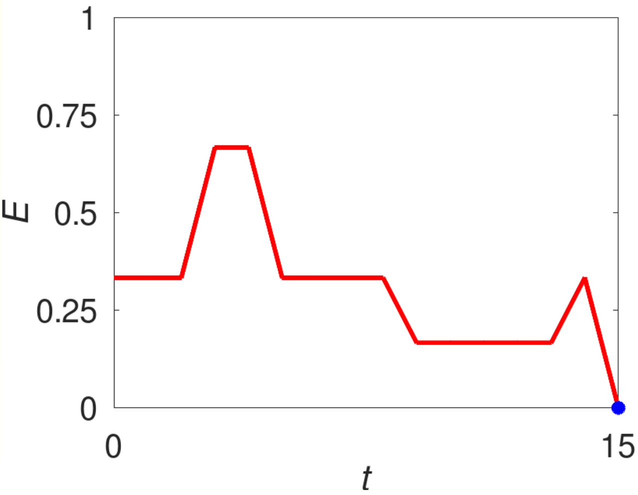
Ex7 $(I = 1)$ | Perc–v1 | $t = 15$

- $\tilde{w}_{(0)} = (0, 0, 0)^\top, E_{(0)} = 0.33333$
- $\tilde{w}^* = (1, 0, -2)^\top, E_{(15)} = 0$



Ex7 $(I = 1)$ | Perc–v1 | E

Ex7 $(I = 1)$ | Perc–v1 | E

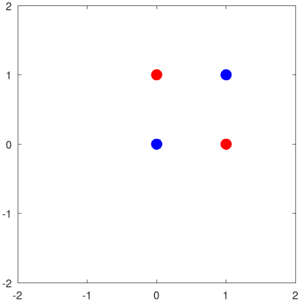


Ex3 (XOR) | D

- Base de dados "XOR": $D = (x^n, y^n)_{n=1}^{4}$ com

$x^1 = (0,0)^\top$ $\quad y^1 = -1$ (F) $\qquad x^2 = (0,1)^\top$ $\quad y^2 = +1$ (V)

$x^3 = (1,0)^\top$ $\quad y^3 = +1$ (V) $\qquad x^4 = (1,1)^\top$ $\quad y^4 = -1$ (F)

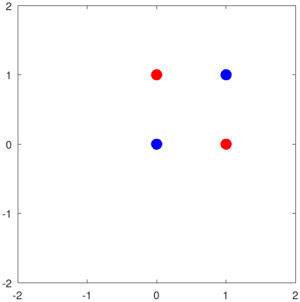


Ex3 (XOR) | D

- Base de dados "XOR": $D = (x^n, y^n)_{n=1}^{4}$ com

  $x^1 = (0,0)^\top \quad y^1 = -1$ (F) $\qquad x^2 = (0,1)^\top \quad y^2 = +1$ (V)

  $x^3 = (1,0)^\top \quad y^3 = +1$ (V) $\qquad x^4 = (1,1)^\top \quad y^4 = -1$ (F)



- D não é linearmente separável.

Ex3 (XOR) | linearização | D' (i)

- Nova base de dados: $D' = (z^n, y^n)_{n=1}^4$ com

$$z^n = \Phi(x^n) = \Phi(x_1^n, x_2^n) = (x_1^n, x_2^n, x_1^n x_2^n)^\top,$$

ou seja,

$$z^1 = \Phi(x^1) = \Phi(0,0) = (0,0,0)^\top \qquad y^1 = -1$$
$$z^2 = \Phi(x^2) = \Phi(0,1) = (0,1,0)^\top \qquad y^2 = +1$$
$$z^3 = \Phi(x^3) = \Phi(1,0) = (1,0,0)^\top \qquad y^3 = +1$$
$$z^4 = \Phi(x^4) = \Phi(1,1) = (1,1,1)^\top \qquad y^4 = -1$$

- D' já é linearmente separável — verificação, por exemplo, com o classificador $\tilde{w} = (-\frac{1}{2}, 1, 1, -2)$, que $y^n(\tilde{w} \cdot \tilde{z}^n) > 0$, $n = 1, 2, 3, 4$:

$$y^1(\tilde{w} \cdot \tilde{z}^1) = (-1) \times ((-\tfrac{1}{2}, 1, 1, -2) \cdot (1, 0, 0, 0)) = \frac{1}{2} > 0,$$
$$y^2(\tilde{w} \cdot \tilde{z}^2) = (+1) \times ((-\tfrac{1}{2}, 1, 1, -2) \cdot (1, 0, 1, 0)) = \frac{1}{2} > 0,$$
$$y^3(\tilde{w} \cdot \tilde{z}^3) = (+1) \times ((-\tfrac{1}{2}, 1, 1, -2) \cdot (1, 1, 0, 0)) = \frac{1}{2} > 0,$$
$$y^4(\tilde{w} \cdot \tilde{z}^4) = (-1) \times ((-\tfrac{1}{2}, 1, 1, -2) \cdot (1, 1, 1, 1)) = \frac{1}{2} > 0.$$

Linearização | ideia

- Dada uma base de dados $D = (x^n, y^n)_{n=1}^N$, $x^n \in \mathbb{R}^I$, $y^n \in \{-1, +1\}$,
- construir uma nova base de dados $D' = (z^n, y^n)_{n=1}^N$, $z^n = \Phi(x^n) \in \mathbb{R}^J$, a partir de D considerando uma *feature map* (transformação nos atributos)

$$\begin{array}{rccc} \Phi : & \mathbb{R}^I & \longrightarrow & \mathbb{R}^{J(>I)} \\ & x & \longmapsto & z = \Phi(x) \end{array}$$

- Chama-se
  - x^n: atributos primários (*input space*: $\mathbb{R}^I$).
  - z^n: atributos secundários (*feature space*: $\mathbb{R}^J$).
- Arquitetura da *Machine Learning*: $h(z; \tilde{w}) = \operatorname{sgn}(\tilde{w} \cdot \tilde{z})$.
- Que *feature maps* considerar? Por exemplo, polinómios de grau d — problema: se, por exemplo, $I = 100$, para $d = 2$ tem-se que $J = 5050$ e para $d = 3$ tem-se que $J = 343\,400$, o que pode tornar inviável este procedimento.
- Será possível evitar o cálculo explícito de $\Phi(x^n)$? Sim, considerando dois novos ingredientes: representação dual e *kernel trick*.

Algoritmo Perc–v1 | atributos primários

Input: $D = (x^n, y^n)_{n=1}^N$, $x^n \in \mathbb{R}^I$, $y^n \in \{-1, +1\}$, $\tilde{w}_{(0)} \in \mathbb{R}^{I+1}$
Output: $\tilde{w}^* \in \mathbb{R}^{I+1}$

$t \leftarrow 0$;
while *V* **do**
    for $n \leftarrow 1$ **to** N **do**
        $E_{(t)} \leftarrow \dfrac{1}{N} \displaystyle\sum_{p=1}^{N} \dfrac{1}{2} \Big| y^p - \underbrace{\operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{x}^p\right)}_{\hat{y}^p} \Big|$;
        if $E_{(t)} = 0$ **then**
            $\tilde{w}^* \leftarrow \tilde{w}_{(t)}$; **return** $\tilde{w}^*$;
        else
            $\hat{y}^n \leftarrow \operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{x}^n\right)$;
            if $y^n \neq \hat{y}^n$ **then**
                $\tilde{w}_{(t+1)} \leftarrow \tilde{w}_{(t)} + y^n \tilde{x}^n$;
            else
                $\tilde{w}_{(t+1)} \leftarrow \tilde{w}_{(t)}$;
            $t \leftarrow t + 1$;

Ex1 (AND) | D

- Base de dados "AND": $D = (x^n, y^n)_{n=1}^4$ com

  $x^1 = (0,0)^\top$ $\quad y^1 = -1$ (F) $\qquad x^2 = (0,1)^\top$ $\quad y^2 = -1$ (F)

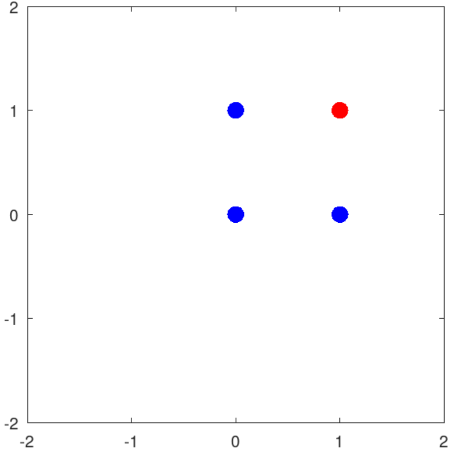
  $x^3 = (1,0)^\top$ $\quad y^3 = -1$ (F) $\qquad x^4 = (1,1)^\top$ $\quad y^4 = +1$ (V)

Ex1 (AND) | D

- Base de dados "AND": $D = (x^n, y^n)_{n=1}^4$ com

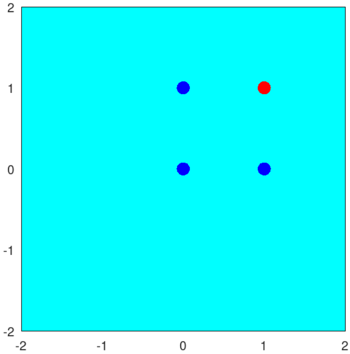
$x^1 = (0,0)^\top \quad y^1 = -1$ (F) $\qquad x^2 = (0,1)^\top \quad y^2 = -1$ (F)

$x^3 = (1,0)^\top \quad y^3 = -1$ (F) $\qquad x^4 = (1,1)^\top \quad y^4 = +1$ (V)



Ex1 (AND) | Perc–v1 | $t = 0$

- $\tilde{w}_{(0)} = (0, 0, 0)^\top, E_{(0)} = 0.25$

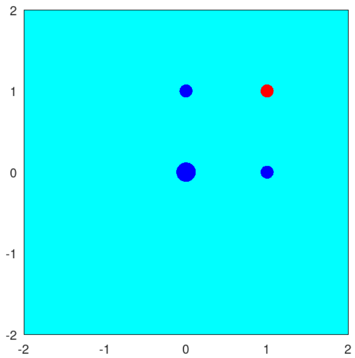


Ex1 (AND) | Perc–v1 | $t = 1$

- $n = 1, \tilde{w}_{(1)} = (0,0,0)^\top, E_{(1)} = 0.25$
- como x^1 é bem classificado por $\tilde{w}_{(0)}$, $\tilde{w}_{(1)} \leftarrow \tilde{w}_{(0)}$, vindo

$$\tilde{w}_{(1)} = \tilde{w}_{(0)} + 0\tilde{x}^1 + 0\tilde{x}^2 + 0\tilde{x}^3 + 0\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(1),\ell}\tilde{x}^\ell$$

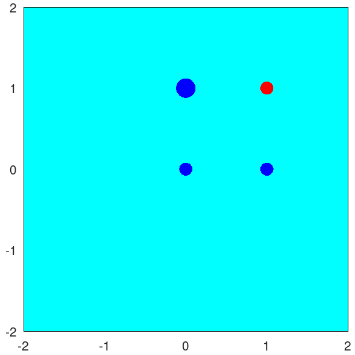
$$\alpha_{(1)} = (0,0,0,0)^\top$$



- $n = 2, \tilde{w}_{(2)} = (0,0,0)^\top, E_{(2)} = 0.25$
- como x^2 é bem classificado por $\tilde{w}_{(1)}$, $\tilde{w}_{(2)} \leftarrow \tilde{w}_{(1)}$, vindo

$$\tilde{w}_{(2)} = \tilde{w}_{(0)} + 0\tilde{x}^1 + 0\tilde{x}^2 + 0\tilde{x}^3 + 0\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(2),\ell}\tilde{x}^\ell$$

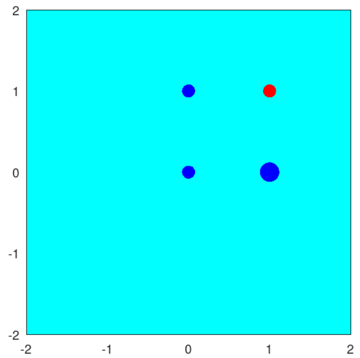
$$\alpha_{(2)} = (0,0,0,0)^\top$$



- $n = 3, \tilde{w}_{(3)} = (0,0,0)^\top, E_{(3)} = 0.25$
- como x^3 é bem classificado por $\tilde{w}_{(2)}$, $\tilde{w}_{(3)} \leftarrow \tilde{w}_{(2)}$, vindo

$$\tilde{w}_{(3)} = \tilde{w}_{(0)} + 0\tilde{x}^1 + 0\tilde{x}^2 + 0\tilde{x}^3 + 0\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(3),\ell}\tilde{x}^\ell$$

$$\alpha_{(3)} = (0,0,0,0)^\top$$

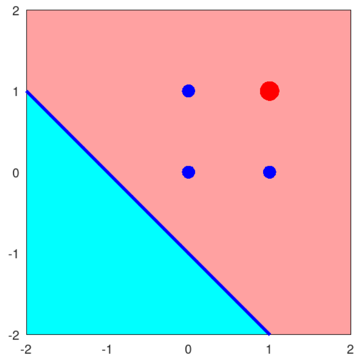


Ex1 (AND) | Perc–v1 | $t = 4$

- $n = 4, \tilde{w}_{(4)} = (1, 1, 1)^\top, E_{(4)} = 0.75$
- como x^4 é mal classificado por $\tilde{w}_{(3)}$, $\tilde{w}_{(4)} \leftarrow \tilde{w}_{(3)} + y^4 \tilde{x}^4$, vindo

$$\tilde{w}_{(4)} = \tilde{w}_{(0)} + 0\tilde{x}^1 + 0\tilde{x}^2 + 0\tilde{x}^3 + 1\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(4),\ell} \tilde{x}^\ell$$

$$\alpha_{(4)} = (0, 0, 0, 1)^\top$$

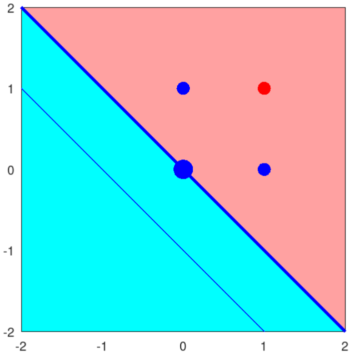


Ex1 (AND) | Perc–v1 | $t = 5$

- $n = 1, \tilde{w}_{(5)} = (0,1,1)^\top, E_{(5)} = 0.5$
- como x^1 é mal classificado por $\tilde{w}_{(4)}$, $\tilde{w}_{(5)} \leftarrow \tilde{w}_{(4)} + y^1\tilde{x}^1$, vindo

$$\tilde{w}_{(5)} = \tilde{w}_{(0)} - 1\tilde{x}^1 + 0\tilde{x}^2 + 0\tilde{x}^3 + 1\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(5),\ell}\tilde{x}^\ell$$

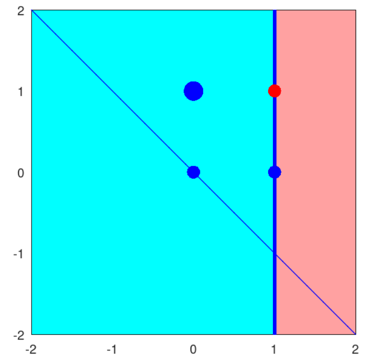
$$\alpha_{(5)} = (-1,0,0,1)^\top$$



- $n = 2, \tilde{w}_{(6)} = (-1, 1, 0)^\top, E_{(6)} = 0.25$
- como x^2 é mal classificado por $\tilde{w}_{(5)}$, $\tilde{w}_{(6)} \leftarrow \tilde{w}_{(5)} + y^2\tilde{x}^2$, vindo

$$\tilde{w}_{(6)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 1\tilde{x}^2 + 0\tilde{x}^3 + 1\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(6),\ell}\tilde{x}^\ell$$

$$\alpha_{(6)} = (-1, -1, 0, 1)^\top$$

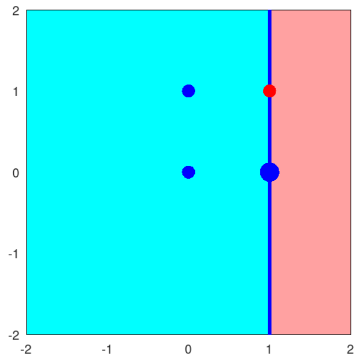


Ex1 (AND) | Perc–v1 | $t = 7$

- $n = 3, \tilde{w}_{(7)} = (-1, 1, 0)^\top, E_{(7)} = 0.25$
- como x^3 é bem classificado por $\tilde{w}_{(6)}$, $\tilde{w}_{(7)} \leftarrow \tilde{w}_{(6)}$, vindo

$$\tilde{w}_{(7)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 1\tilde{x}^2 + 0\tilde{x}^3 + 1\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(7),\ell}\tilde{x}^\ell$$

$$\alpha_{(7)} = (-1, -1, 0, 1)^\top$$

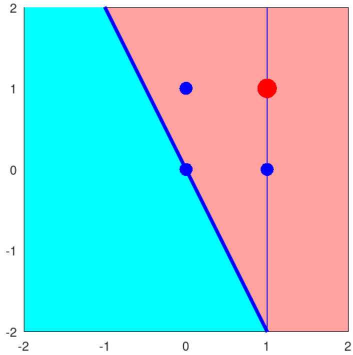


Ex1 (AND) | Perc–v1 | $t = 8$

- $n = 4, \tilde{w}_{(8)} = (0, 2, 1)^\top, E_{(8)} = 0.5$
- como x^4 é mal classificado por $\tilde{w}_{(7)}$, $\tilde{w}_{(8)} \leftarrow \tilde{w}_{(7)} + y^4 \tilde{x}^4$, vindo

$$\tilde{w}_{(8)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 1\tilde{x}^2 + 0\tilde{x}^3 + 2\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(8),\ell} \tilde{x}^\ell$$

$$\alpha_{(8)} = (-1, -1, 0, 2)^\top$$

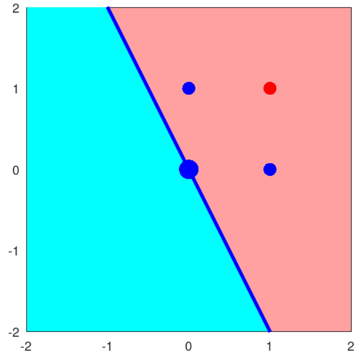


Ex1 (AND) | Perc–v1 | $t = 9$

- $n = 1, \tilde{w}_{(9)} = (0, 2, 1)^\top, E_{(9)} = 0.5$
- como x^1 é bem classificado por $\tilde{w}_{(8)}$, $\tilde{w}_{(9)} \leftarrow \tilde{w}_{(8)}$, vindo

$$\tilde{w}_{(9)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 1\tilde{x}^2 + 0\tilde{x}^3 + 2\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(9),\ell}\tilde{x}^\ell$$

$$\alpha_{(9)} = (-1, -1, 0, 2)^\top$$

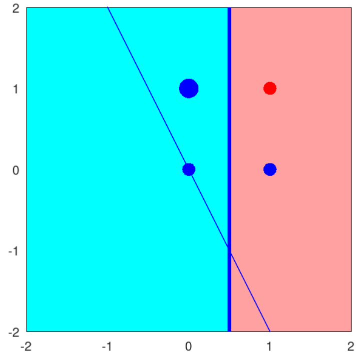


Ex1 (AND) | Perc–v1 | $t = 10$

- $n = 2, \tilde{w}_{(10)} = (-1, 2, 0)^\top, E_{(10)} = 0.25$
- como x^2 é mal classificado por $\tilde{w}_{(9)}$, $\tilde{w}_{(10)} \leftarrow \tilde{w}_{(9)} + y^2\tilde{x}^2$, vindo

$$\tilde{w}_{(10)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 + 0\tilde{x}^3 + 2\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(10),\ell}\tilde{x}^\ell$$

$$\alpha_{(10)} = (-1, -2, 0, 2)^\top$$

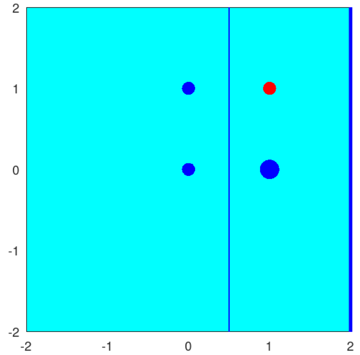


Ex1 (AND) | Perc–v1 | $t = 11$

- $n = 3, \tilde{w}_{(11)} = (-2, 1, 0)^\top, E_{(11)} = 0.25$
- como x^3 é mal classificado por $\tilde{w}_{(10)}$, $\tilde{w}_{(11)} \leftarrow \tilde{w}_{(10)} + y^3 \tilde{x}^3$, vindo

$$\tilde{w}_{(11)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 1\tilde{x}^3 + 2\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(11),\ell} \tilde{x}^\ell$$

$$\alpha_{(11)} = (-1, -2, -1, 2)^\top$$

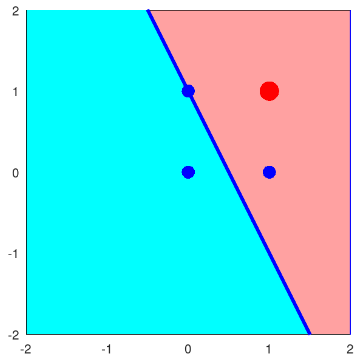


Ex1 (AND) | Perc–v1 | $t = 12$

- $n = 4, \tilde{w}_{(12)} = (-1, 2, 1)^\top, E_{(12)} = 0.25$
- como x^4 é mal classificado por $\tilde{w}_{(11)}$, $\tilde{w}_{(12)} \leftarrow \tilde{w}_{(11)} + y^4 \tilde{x}^4$, vindo

$$\tilde{w}_{(12)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 1\tilde{x}^3 + 3\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(12),\ell}\tilde{x}^\ell$$

$$\alpha_{(12)} = (-1, -2, -1, 3)^\top$$

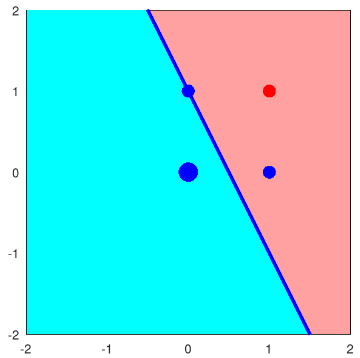


Ex1 (AND) | Perc–v1 | $t = 13$

- $n = 1, \tilde{w}_{(13)} = (-1, 2, 1)^\top, E_{(13)} = 0.25$
- como x^1 é bem classificado por $\tilde{w}_{(12)}$, $\tilde{w}_{(13)} \leftarrow \tilde{w}_{(12)}$, vindo

$$\tilde{w}_{(13)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 1\tilde{x}^3 + 3\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(13),\ell}\tilde{x}^\ell$$

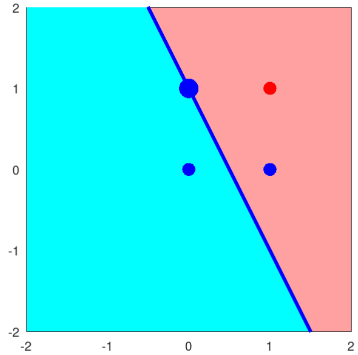
$$\alpha_{(13)} = (-1, -2, -1, 3)^\top$$



- $n = 2, \tilde{w}_{(14)} = (-1, 2, 1)^\top, E_{(14)} = 0.25$
- como x^2 é bem classificado por $\tilde{w}_{(13)}$, $\tilde{w}_{(14)} \leftarrow \tilde{w}_{(13)}$, vindo

$$\tilde{w}_{(14)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 1\tilde{x}^3 + 3\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(14),\ell}\tilde{x}^\ell$$

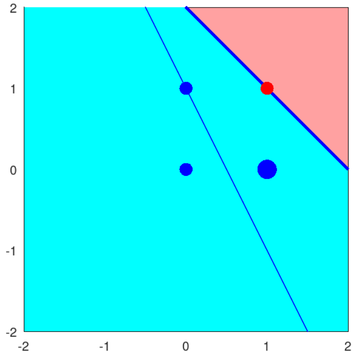
$$\alpha_{(14)} = (-1, -2, -1, 3)^\top$$



- $n = 3, \tilde{w}_{(15)} = (-2, 1, 1)^\top, E_{(15)} = 0.25$
- como x^3 é mal classificado por $\tilde{w}_{(14)}$, $\tilde{w}_{(15)} \leftarrow \tilde{w}_{(14)} + y^3\tilde{x}^3$, vindo

$$\tilde{w}_{(15)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 2\tilde{x}^3 + 3\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(15),\ell}\tilde{x}^\ell$$

$$\alpha_{(15)} = (-1, -2, -2, 3)^\top$$

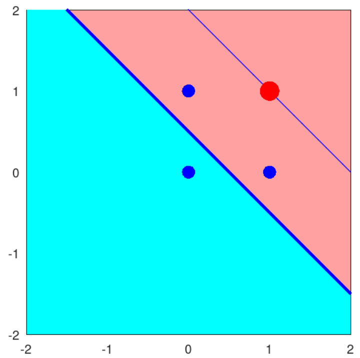


Ex1 (AND) | Perc–v1 | $t = 16$

- $n = 4, \tilde{w}_{(16)} = (-1, 2, 2)^\top, E_{(16)} = 0.5$
- como x^4 é mal classificado por $\tilde{w}_{(15)}$, $\tilde{w}_{(16)} \leftarrow \tilde{w}_{(15)} + y^4 \tilde{x}^4$, vindo

$$\tilde{w}_{(16)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 2\tilde{x}^3 + 4\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(16),\ell} \tilde{x}^\ell$$

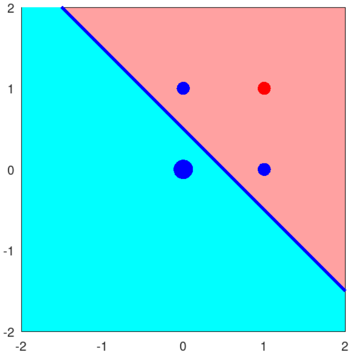
$$\alpha_{(16)} = (-1, -2, -2, 4)^\top$$



- $n = 1, \tilde{w}_{(17)} = (-1, 2, 2)^\top, E_{(17)} = 0.5$
- como x^1 é bem classificado por $\tilde{w}_{(16)}$, $\tilde{w}_{(17)} \leftarrow \tilde{w}_{(16)}$, vindo

$$\tilde{w}_{(17)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 2\tilde{x}^2 - 2\tilde{x}^3 + 4\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(17),\ell}\tilde{x}^\ell$$

$$\alpha_{(17)} = (-1, -2, -2, 4)^\top$$

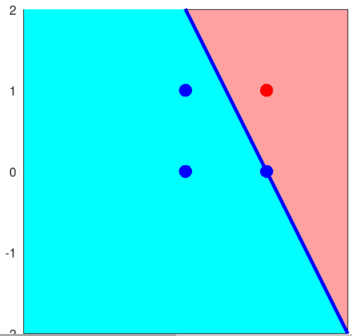


Ex1 (AND) | Perc–v1 | $t = 18$

- $\tilde{w}_{(0)} = (0,0,0)^\top, E_{(0)} = 0.25$
- $\tilde{w}^* = (-2,2,1)^\top, E_{(18)} = 0$
- como x^2 é mal classificado por $\tilde{w}_{(17)}$, $\tilde{w}_{(18)} \leftarrow \tilde{w}_{(17)} + y^2\tilde{x}^2$, vindo

$$\tilde{w}_{(18)} = \tilde{w}_{(0)} - 1\tilde{x}^1 - 3\tilde{x}^2 - 2\tilde{x}^3 + 4\tilde{x}^4 = \tilde{w}_{(0)} + \sum_{\ell=1}^{4} \alpha_{(18),\ell}\tilde{x}^\ell$$

$$\alpha_{(18)} = (-1,-3,-2,4)^\top$$



Ex1 (AND) | Perc–v1

Ex1 (AND) | Perc–v1

- O algoritmo convergiu em 18 iterações com

Ex1 (AND) | Perc–v1

- O algoritmo convergiu em 18 iterações com
  - $\tilde{w}^* = \tilde{w}_{(18)} = (-2, 2, 1)^\top$,

Ex1 (AND) | Perc–v1

- O algoritmo convergiu em 18 iterações com
  - $\tilde{w}^* = \tilde{w}_{(18)} = (-2, 2, 1)^\top$,
  - $\alpha^* = \alpha_{(18)} = (-1, -3, -2, 4)^\top$.

Ex1 (AND) | Perc–v1

- O algoritmo convergiu em 18 iterações com
  - $\tilde{w}^* = \tilde{w}_{(18)} = (-2, 2, 1)^\top$,
  - $\alpha^* = \alpha_{(18)} = (-1, -3, -2, 4)^\top$.
- O vetor dos pesos é uma combinação linear dos vetores dos inputs a menos de $\tilde{w}_{(0)}$

$$\tilde{w}^* = \tilde{w}_{(0)} + \sum_{\ell=1}^{N} \alpha_\ell^* \tilde{x}^\ell.$$

Ex1 (AND) | Perc–v1

- O algoritmo convergiu em 18 iterações com
  - $\tilde{w}^* = \tilde{w}_{(18)} = (-2, 2, 1)^\top$,
  - $\alpha^* = \alpha_{(18)} = (-1, -3, -2, 4)^\top$.
- O vetor dos pesos é uma combinação linear dos vetores dos inputs a menos de $\tilde{w}_{(0)}$

$$\tilde{w}^* = \tilde{w}_{(0)} + \sum_{\ell=1}^{N} \alpha_\ell^* \tilde{x}^\ell.$$

- $|\alpha_\ell^*|$ é igual ao número de vezes que x^ℓ foi mal classificado ao longo do processo iterativo.

Ex1 (AND) | Perc–v1

- Verificação da expressão $\tilde{w}^* = \tilde{w}_{(0)} + \sum_{\ell=1}^{N} \alpha_\ell^* \tilde{x}^\ell$:

$$
\begin{aligned}
&x^1 = (0,0)^\top, x^2 = (0,1)^\top, x^3 = (1,0)^\top, x^4 = (1,1)^\top \\
&\tilde{w}_{(0)} = (0,0,0)^\top \\
&\tilde{w}^* = (-2,2,1)^\top \\
&\alpha^* = (-1,-3,-2,4)^\top
\end{aligned}
$$

$$
\begin{aligned}
\tilde{w}^* &= \tilde{w}_{(0)} + \sum_{\ell=1}^{N} \alpha_\ell^* \tilde{x}^\ell \\
&= \tilde{w}_{(0)} + \alpha_1^* \tilde{x}^1 + \alpha_2^* \tilde{x}^2 + \alpha_3^* \tilde{x}^3 + \alpha_4^* \tilde{x}^4 \\
&= (0,0,0)^\top - 1(1,0,0)^\top - 3(1,0,1)^\top - 2(1,1,0)^\top + 4(1,1,1)^\top \\
&= (-2,2,1)^\top.
\end{aligned}
$$

- A partir de agora vai-se considerar que $\tilde{w}_{(0)}$ é o vetor nulo, pelo que

$$\tilde{w} = \sum_{\ell=1}^{N} \alpha_\ell \tilde{x}^\ell$$

Perceptron | versão dual

- A partir de agora vai-se considerar que $\tilde{w}_{(0)}$ é o vetor nulo, pelo que

$$\tilde{w} = \sum_{\ell=1}^{N} \alpha_\ell \tilde{x}^\ell$$

- Arquitetura da *Machine Learning*

$$\begin{aligned} h(x; \tilde{w}) &= \operatorname{sgn}(\tilde{w} \cdot \tilde{x}) \\ &= \operatorname{sgn}\left(\left(\sum_{\ell=1}^{N} \alpha_\ell \tilde{x}^\ell\right) \cdot \tilde{x}\right) \\ &= \operatorname{sgn}\left(\sum_{\ell=1}^{N} \alpha_\ell \left(\tilde{x}^\ell \cdot \tilde{x}\right)\right). \end{aligned}$$

Perceptron | versão dual

- A partir de agora vai-se considerar que $\tilde{w}_{(0)}$ é o vetor nulo, pelo que

$$\tilde{w} = \sum_{\ell=1}^{N} \alpha_\ell \tilde{x}^\ell$$

- Arquitetura da *Machine Learning*

$$\begin{aligned} h(x;\tilde{w}) &= \operatorname{sgn}(\tilde{w}\cdot\tilde{x}) \\ &= \operatorname{sgn}\left(\left(\sum_{\ell=1}^{N} \alpha_\ell \tilde{x}^\ell\right)\cdot\tilde{x}\right) \\ &= \operatorname{sgn}\left(\sum_{\ell=1}^{N} \alpha_\ell \left(\tilde{x}^\ell\cdot\tilde{x}\right)\right). \end{aligned}$$

- Para determinar

$$\alpha^* = \left(\operatorname*{arg\,min}_{\alpha\in\mathbb{R}^N} E(\tilde{w}, D)\right) \in \mathbb{R}^N,$$

constrói-se uma sequência $\left(\alpha_{(t)}\right)_{t\in\mathbb{N}_0}$ tal que $\alpha_{(t)} \to \alpha^*$.

Algoritmo PercDual–v1

Input: $D = (x^n, y^n)_{n=1}^N$, $x^n \in \mathbb{R}^I$, $y^n \in \{-1, +1\}$
Output: $\alpha^* \in \mathbb{R}^N$

$t \leftarrow 0$;
$\alpha_{(0)} \leftarrow (0, \ldots, 0)^\top \in \mathbb{R}^N$;
while *V* **do**
  for $n \leftarrow 1$ **to** N **do**
    $E_{(t)} \leftarrow \frac{1}{N} \sum_{p=1}^N \frac{1}{2} \left| y^p - \underbrace{\operatorname{sgn}\left(\sum_{\ell=1}^N \alpha_{(t),\ell} \left(\tilde{x}^\ell \cdot \tilde{x}^n \right) \right)}_{\hat{y}^p} \right|$;
    if $E_{(t)} = 0$ **then**
      $\alpha^* \leftarrow \alpha_{(t)}$; **return** α^*;
    else
      $\hat{y}^n \leftarrow \operatorname{sgn}\left(\sum_{\ell=1}^N \alpha_{(t),\ell} \left(\tilde{x}^\ell \cdot \tilde{x}^n \right) \right)$;
      if $y^n \neq \hat{y}^n$ **then**
        $\alpha_{(t+1),n} \leftarrow \alpha_{(t),n} + y^n$;
        $\alpha_{(t+1),\ell} \leftarrow \alpha_{(t),\ell}$, $\ell = 1, \ldots, n-1, n+1, \ldots, N$;
      else
        $\alpha_{(t+1)} \leftarrow \alpha_{(t)}$;
      $t \leftarrow t + 1$;

Perceptron | algoritmo dual

- Vai-se chamar "versão primal" quando o classificador é dado através do vetor $\tilde{w}$ e "versão dual" quando o classificador é dado através do vetor α.

Perceptron | algoritmo dual

- Vai-se chamar "versão primal" quando o classificador é dado através do vetor $\tilde{w}$ e "versão dual" quando o classificador é dado através do vetor α.
- No slide anterior apresentou-se a versão dual do algoritmo Perc–v1, que se denota por PercDual–v1, sendo também possível construir versões duais para todas as versões primais.

- Vai-se chamar "versão primal" quando o classificador é dado através do vetor $\tilde{w}$ e "versão dual" quando o classificador é dado através do vetor α.
- No slide anterior apresentou-se a versão dual do algoritmo Perc–v1, que se denota por PercDual–v1, sendo também possível construir versões duais para todas as versões primais.
- Se

$$\tilde{w}_{(0)} = (0, \ldots, 0) \in \mathbb{R}^{I+1} \text{ e}$$

$$\alpha_{(0)} = (0, \ldots, 0) \in \mathbb{R}^{N},$$

os resultados dos algoritmos Perc–v1 e PercDual–v1 são, iteração a iteração, iguais.

- Recorde-se que o objetivo é tratar bases de dados que não são linearmente separáveis através do modelo linear *Perceptron*, aumentando para tal a dimensão do espaço dos atributos.

Kernel | objetivo

- Recorde-se que o objetivo é tratar bases de dados que não são linearmente separáveis através do modelo linear *Perceptron*, aumentando para tal a dimensão do espaço dos atributos.
- A ideia é o algoritmo trabalhar numa nova base de dados $D' = (z^n, y^n)_{n=1}^N$, $z^n = \Phi(x^n) \in \mathbb{R}^J$, a partir de D considerando uma *feature map* (transformação nos atributos)

$$\begin{array}{rccc} \Phi : & \mathbb{R}^I & \longrightarrow & \mathbb{R}^{J(>I)} \\ & x & \longmapsto & z = \Phi(x), \end{array}$$

mas evitando o cálculo explícito de $\Phi(x^n)$.

- Recorde-se que o objetivo é tratar bases de dados que não são linearmente separáveis através do modelo linear *Perceptron*, aumentando para tal a dimensão do espaço dos atributos.
- A ideia é o algoritmo trabalhar numa nova base de dados $D' = (z^n, y^n)_{n=1}^N$, $z^n = \Phi(x^n) \in \mathbb{R}^J$, a partir de D considerando uma *feature map* (transformação nos atributos)

$$\begin{array}{rccc} \Phi: & \mathbb{R}^I & \longrightarrow & \mathbb{R}^{J(>I)} \\ & x & \longmapsto & z = \Phi(x), \end{array}$$

  mas evitando o cálculo explícito de $\Phi(x^n)$.
- O objetivo é remover do algoritmo a dependência explícita de $\tilde{z}$ via $\tilde{\Phi}$ (linhas 4 e 6 do algoritmo versão v1 do algoritmo *Perceptron*)

$$\begin{array}{rccc} \tilde{\Phi}: & \mathbb{R}^{I+1} & \longrightarrow & \mathbb{R}^{J+1} \\ & \tilde{x} & \longmapsto & \tilde{z} = \tilde{\Phi}(\tilde{x}). \end{array}$$

Algoritmo Perc–v1 | atributos secundários

Input: $D' = (z^n, y^n)_{n=1}^N$, $z^n = \Phi(x^n) \in \mathbb{R}^J$, $y^n \in \{-1, +1\}$, $\tilde{w}_{(0)} \in \mathbb{R}^{J+1}$
Output: $\tilde{w}^* \in \mathbb{R}^{J+1}$

$t \leftarrow 0$;
while V **do**
    for $n \leftarrow 1$ **to** N **do**
        $E_{(t)} \leftarrow \dfrac{1}{N} \sum_{p=1}^{N} \dfrac{1}{2} \Big| y^p - \underbrace{\operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{z}^p\right)}_{\hat{y}^p} \Big|$;
        if $E_{(t)} = 0$ **then**
            $\tilde{w}^* \leftarrow \tilde{w}_{(t)}$; **return** $\tilde{w}^*$;
        else
            $\hat{y}^n \leftarrow \operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{z}^n\right)$;
            if $y^n \neq \hat{y}^n$ **then**
                $\tilde{w}_{(t+1)} \leftarrow \tilde{w}_{(t)} + y^n \tilde{z}^n$;
            else
                $\tilde{w}_{(t+1)} \leftarrow \tilde{w}_{(t)}$;
            $t \leftarrow t + 1$;

- Considerando a representação dual do algoritmo *Perceptron*
  - a linha 6 $\boxed{\tilde{w}_{(t+1)} \leftarrow \tilde{w}_{(t)} + y^n \tilde{z}^n}$ é substituída pela atualização do vetor α

$$\alpha_{(t+1),n} \leftarrow \alpha_{(t),n} + y^n$$
$$\alpha_{(t+1),\ell} \leftarrow \alpha_{(t),\ell}, \ell = 1, \ldots, n-1, n+1, \ldots, N$$

  - a linha 4 $\boxed{\hat{y}^n \leftarrow \operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{z}^n\right)}$, atendendo a,

$$\tilde{w} = \sum_{\ell=1}^{N} \alpha_\ell \tilde{\Phi}(\tilde{x}^\ell) = \sum_{\ell=1}^{N} \alpha_\ell \tilde{z}^\ell$$

pode ser reescrita como

$$\begin{aligned}\hat{y}^n \leftarrow \operatorname{sgn}\left(\tilde{w}_{(t)} \cdot \tilde{z}^n\right) &= \operatorname{sgn}\left(\left(\sum_{\ell=1}^{N} \alpha_{(t),\ell} \tilde{z}^\ell\right) \cdot \tilde{z}^n\right) \\ &= \operatorname{sgn}\left(\sum_{\ell=1}^{N} \alpha_{(t),\ell} \left(\tilde{\Phi}(\tilde{x}^\ell) \cdot \tilde{\Phi}(\tilde{x}^n)\right)\right),\end{aligned}$$

pelo que, se existir uma função K tal que $K(x, x') = \tilde{\Phi}(x) \cdot \tilde{\Phi}(x')$, o problema fica resolvido.

Kernel | exemplo com $I = 2$ e grau $d = 2$

- **Exemplo.** Seja a *feature map* com $I = 2$ e grau $d = 2$:

$$\begin{array}{cccc} \tilde{\Phi}: & \mathbb{R}^{2+1} & \longrightarrow & \mathbb{R}^{5+1} \\ & \tilde{x} = (1, x_1, x_2) & \longmapsto & \tilde{z} = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right) \end{array}$$

$$\begin{aligned}
\tilde{\Phi}(\tilde{x}) \cdot \tilde{\Phi}(\tilde{x}') &= \tilde{\Phi}(1, x_1, x_2) \cdot \tilde{\Phi}(1, x_1', x_2') \\
&= \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right) \cdot \left(1, \sqrt{2}x_1', \sqrt{2}x_2', (x_1')^2, \sqrt{2}x_1'x_2', (x_2')^2\right) \\
&= 1 + 2x_1x_1' + 2x_2x_2' + (x_1)^2(x_1')^2 + 2x_1x_1'x_2x_2' + (x_2)^2(x_2')^2 \\
&= (1 + x_1x_1' + x_2x_2')^2 \\
&= ((1, x_1, x_2) \cdot (1, x_1', x_2'))^2 \\
&= (\tilde{x} \cdot \tilde{x}')^2
\end{aligned}$$

- O produto interno em $\mathbb{R}^{J+1}$ pode ser calculado como um produto interno em $\mathbb{R}^{I+1}$ + uma transformação (neste caso, elevando ao quadrado)

$$\tilde{\Phi}(\tilde{x}) \cdot \tilde{\Phi}(\tilde{x}') = (\tilde{x} \cdot \tilde{x}')^2.$$

- *Kernel* polinomial de grau 2: $K(x, x') = (\tilde{x} \cdot \tilde{x}')^2$.

- **Exemplo.** Seja a *feature map* com $I = 2$ e grau $d = 3$:

$$\begin{array}{rccc} \tilde{\Phi}: & \mathbb{R}^{2+1} & \longrightarrow & \mathbb{R}^{9+1} \\ & \tilde{x} = (1, x_1, x_2) & \longmapsto & \tilde{z} = \big(1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{6}x_1 x_2, \sqrt{3}x_2^2, \\ & & & x_1^3, \sqrt{3}x_1^2 x_2, \sqrt{3}x_1 x_2^2, \sqrt{3}x_2^3\big) \end{array}$$

$$\begin{aligned} \tilde{\Phi}(\tilde{x}) \cdot \tilde{\Phi}(\tilde{x}') &= \tilde{\Phi}(1, x_1, x_2) \cdot \tilde{\Phi}(1, x_1', x_2') \\ &= \Big(1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{6}x_1 x_2, \sqrt{3}x_2^2, x_1^3, \sqrt{3}x_1^2 x_2, \sqrt{3}x_1 x_2^2, \sqrt{3}x_2^3\Big) \cdot \\ &\quad \Big(1, \sqrt{3}x_1', \sqrt{3}x_2', \sqrt{3}(x_1')^2, \sqrt{6}x_1' x_2', \sqrt{3}(x_2')^2, (x_1')^3, \sqrt{3}(x_1')^2 x_2', \sqrt{3}x_1'(x_2')^2, \sqrt{3}(x_2')^3\Big) \\ &= 1 + 3x_1 x_1' + 3x_2 x_2' + 3x_1^2(x_1')^2 + 6x_1 x_2 x_1' x_2' + 3x_2^2(x_2')^2 + \\ &\quad x_1^3(x_1')^3 + 3x_1^2 x_2 (x_1')^2 x_2' + 3x_1 x_2^2 x_1'(x_2')^2 + 3x_2^3(x_2')^3 \\ &= (1 + x_1 x_1' + x_2 x_2')^3 \\ &= ((1, x_1, x_2) \cdot (1, x_1', x_2'))^3 \\ &= (\tilde{x} \cdot \tilde{x}')^3 \end{aligned}$$

- O produto interno em $\mathbb{R}^{J+1}$ pode ser calculado como um produto interno em $\mathbb{R}^{I+1}$ + uma transformação (neste caso, elevando ao cubo)

$$\tilde{\Phi}(\tilde{x}) \cdot \tilde{\Phi}(\tilde{x}') = (\tilde{x} \cdot \tilde{x}')^3.$$

- *Kernel* polinomial de grau 3: $K(x, x') = (\tilde{x} \cdot \tilde{x}')^3$.

Kernel | exemplos

- *Kernel* polinomial de grau d

$$K(x,x') = (\tilde{x} \cdot \tilde{x}')^d, x, x' \in \mathbb{R}^I, d \in \mathbb{N}$$

(o caso $d = 1$ corresponde a não fazer nenhuma transformação dos dados originais).

- *Kernel* Gaussiano RBF (*radial basis function*)

$$K(x,x') = \exp(-\gamma \|x - x'\|^2), x, x' \in \mathbb{R}^I, \gamma > 0.$$

- *Kernel* sigmóide

$$K(x,x') = \tanh(\kappa(\tilde{x} \cdot \tilde{x}')), x, x' \in \mathbb{R}^I, \kappa > 0.$$

Kernel | resumo

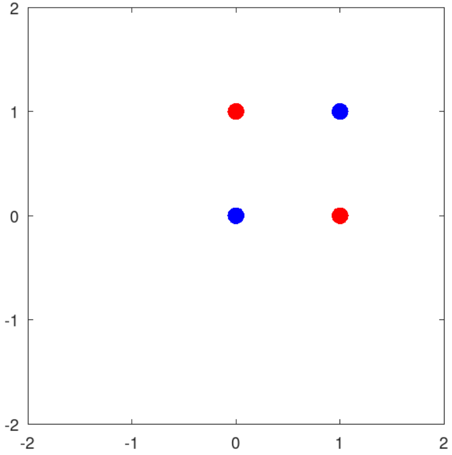
- Por um lado é necessário trabalhar no espaço aumentado $\mathbb{R}^{J+1}$ por forma a obter a separabilidade linear.
- Por outro lado é necessário calcular o produto interno no espaço aumentado, o que pode ser computacionalmente custoso (memória e tempo de cálculo)
- *Kernel trick*: calcular o produto interno no espaço transformado (atributos secundários, J) recorrendo apenas ao espaço original (atributos primários, I).
- Só com a versão dual se consegue aplicar o *kernel trick*.
- No slide seguinte apresenta-se a primeira versão do algoritmo *Perceptron* com a versão dual com *Kernel* polinomial de grau d, que se denota por PercKerPd–v1.

Algoritmo PercKerPd–v1

Input: $D = (x^n, y^n)_{n=1}^N$, $x^n \in \mathbb{R}^I$, $y^n \in \{-1, +1\}$, $\alpha_{(0)} \in \mathbb{R}^N$, $d \in \mathbb{N}$
Output: $\alpha^* \in \mathbb{R}^N$

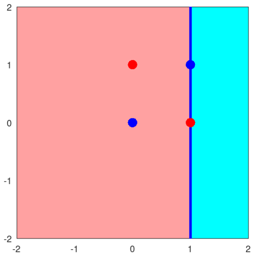
$t \leftarrow 0$;
while V **do**
  for $n \leftarrow 1$ **to** N **do**
    $E_{(t)} \leftarrow \frac{1}{N} \sum_{p=1}^N \frac{1}{2} \left| y^p - \underbrace{\operatorname{sgn}\left(\sum_{\ell=1}^N \alpha_{(t),\ell} \left(\tilde{x}^\ell \cdot \tilde{x}^p \right)^d \right)}_{\hat{y}^p} \right|$;
    if $E_{(t)} = 0$ **then**
      $\alpha^* \leftarrow \alpha_{(t)}$; **return** α^*;
    else
      $\hat{y}^n \leftarrow \operatorname{sgn}\left(\sum_{\ell=1}^N \alpha_{(t),\ell} \left(\tilde{x}^\ell \cdot \tilde{x}^n \right)^d \right)$;
      if $y^n \neq \hat{y}^n$ **then**
        $\alpha_{(t+1),n} \leftarrow \alpha_{(t),n} + y^n$;
        $\alpha_{(t+1),\ell} \leftarrow \alpha_{(t),\ell}$, $\ell = 1, \ldots, n-1, n+1, \ldots, N$;
      else
        $\alpha_{(t+1)} \leftarrow \alpha_{(t)}$;
      $t \leftarrow t + 1$;

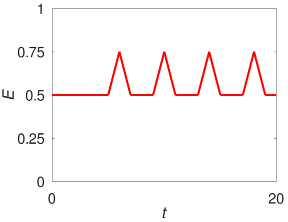
Ex3 (XOR) | D



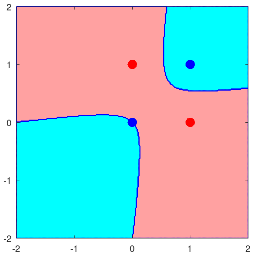
Ex3 (XOR) | E

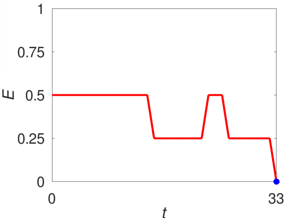
Perc–v2



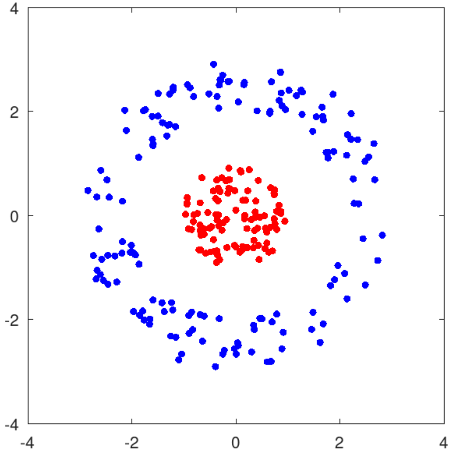


PercKerP2–v1

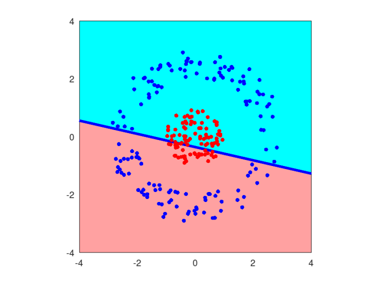


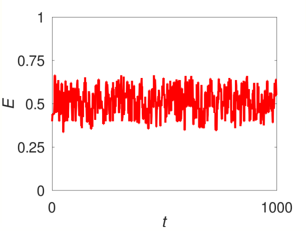


Ex8 | D

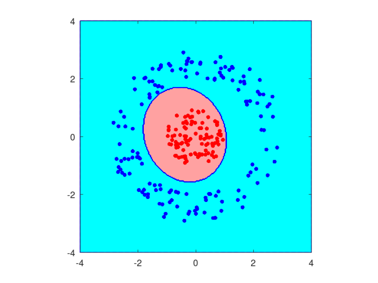


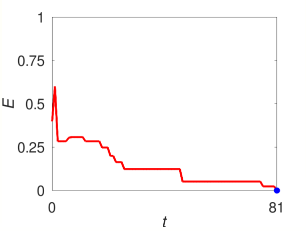
Perc–v2



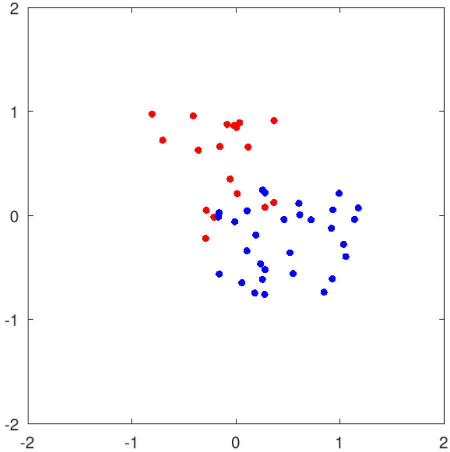


PercKerP2–v1



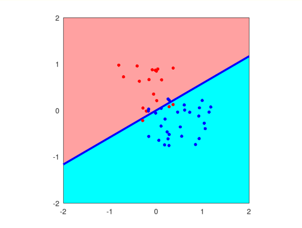


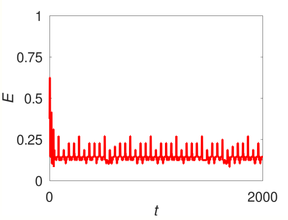
Ex6 | D



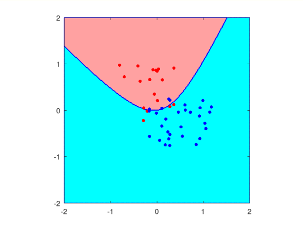
Ex6 | E

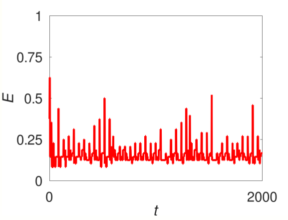
Perc–v2



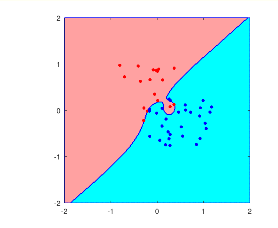


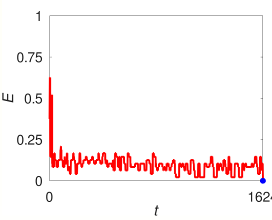
PercKerP2–v2





PercKerP10–v1





Exercícios

Exercício 1. Considere a base de dados binária $D = (x^n, y^n)_{n=1}^{6}$ $(I = 2)$ com

$$
\begin{aligned}
x^1 &= (-1, 1)^\top & y^1 &= -1 \\
x^2 &= (-1, -1)^\top & y^2 &= +1 \\
x^3 &= (0, 0)^\top & y^3 &= +1 \\
x^4 &= (1, 1)^\top & y^4 &= +1 \\
x^5 &= (-1, 0)^\top & y^5 &= -1 \\
x^6 &= (1, -1)^\top & y^6 &= -1
\end{aligned}
$$

Aplique o algoritmo PercKerP2–v2 com $\alpha_{(0)} = (0, 0, 0, 0, 0, 0)^\top$, $T = 2$ e $\eta = 0.5$ à base de dados D, indicando a *accuracy* que se obteve.

Exercícios

Exercício 2. Implemente o algoritmo PercDual–v1 e aplique-o aos exemplos 1 a 6 do capítulo anterior.

Exercício 3. Implemente o algoritmo PercDual–v2 e aplique-o aos exemplos 1 a 6 do capítulo anterior.

Exercício 4. Implemente o algoritmo PercDual–v3 e aplique-o aos exemplos 1 a 6 do capítulo anterior.

Exercício 5.

(a) Construa uma base de dados D com $I = 2$ e com as seguintes características:

- 100 ocorrências com *label* $+1$ que seguem uma distribuição uniforme na região $x_1^2 + x_2^2 + x_3^2 \leqslant r_+^2$, com $r_+ = 1$;
- 150 ocorrências com *label* -1 que seguem uma distribuição uniforme na região $r_{-,1}^2 \leqslant x_1^2 + x_2^2 + x_3^2 \leqslant r_{-,2}^2$, com $r_{-,1} = 2$ e $r_{-,2} = 3$.

(b) Faça o *shuffle* da base de dados.

(c) Treine a *Machine Learning* considerando o algoritmo PercKerP2–v3 e calcule o valor da função custo ao longo das iterações.