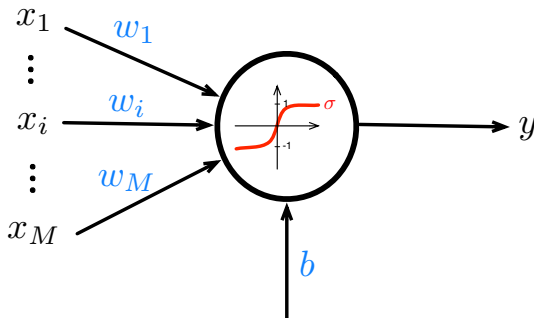


LE PERCEPTRON

Jérémie Cabessa
Laboratoire DAVID, UVSQ

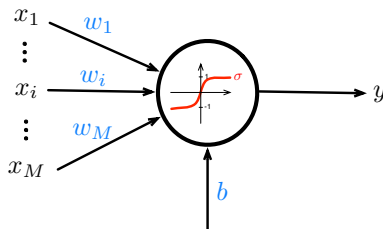
PERCEPTRON

- Le **perceptron** [Rosenblatt, 1957, Rosenblatt, 1958] est un simple neurone qui agit comme un *classifieur binaire*.



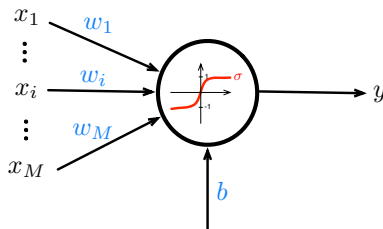
PERCEPTRON

- ▶ $\mathbf{x} = (x_1, \dots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- ▶ $\mathbf{w} = (w_1, \dots, w_M) \in \mathbb{R}^M$ et $b \in \mathbb{R}$ sont les *paramètres*: *poids synaptiques* et *biais*, respectivement.
- ▶ $y \in \{-1, +1\}$ est l'*output* (binaire).
- ▶ σ est la *fonction d'activation*.



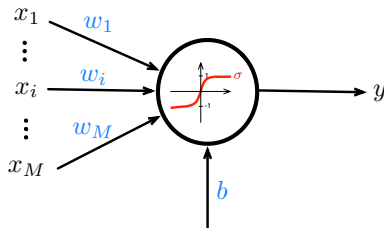
PERCEPTRON

- ▶ $\mathbf{x} = (x_1, \dots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- ▶ $\mathbf{w} = (w_1, \dots, w_M) \in \mathbb{R}^M$ et $b \in \mathbb{R}$ sont les *paramètres*: *poids synaptiques* et *biais*, respectivement.
- ▶ $y \in \{-1, +1\}$ est l'*output* (binaire).
- ▶ σ est la *fonction d'activation*.



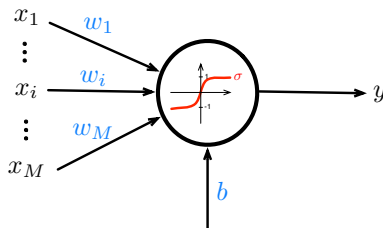
PERCEPTRON

- ▶ $\mathbf{x} = (x_1, \dots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- ▶ $\mathbf{w} = (w_1, \dots, w_M) \in \mathbb{R}^M$ et $b \in \mathbb{R}$ sont les *paramètres*: *poids synaptiques* et *biais*, respectivement.
- ▶ $y \in \{-1, +1\}$ est l'*output* (binaire).
- ▶ σ est la *fonction d'activation*.



PERCEPTRON

- ▶ $\mathbf{x} = (x_1, \dots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- ▶ $\mathbf{w} = (w_1, \dots, w_M) \in \mathbb{R}^M$ et $b \in \mathbb{R}$ sont les *paramètres*: *poids synaptiques* et *biais*, respectivement.
- ▶ $y \in \{-1, +1\}$ est l'*output* (binaire).
- ▶ σ est la *fonction d'activation*.

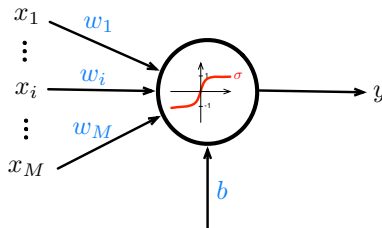


PERCEPTRON

- La dynamique du perceptron est la suivante:

$$y = \sigma(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1, & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1, & \text{if } \mathbf{w}^T \mathbf{x} + b < 0. \end{cases}$$

où $\mathbf{x} = (x_1, \dots, x_M)$ et $\mathbf{w} = (w_1, \dots, w_M)$.



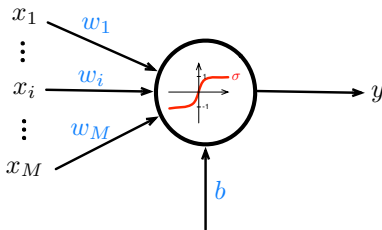
PERCEPTRON

- ▶ Par abus de langage, posons:

$$\mathbf{x} := (1, x_1, \dots, x_M) \text{ et } \mathbf{w} := (b, w_1, \dots, w_M).$$

- ▶ La dynamique du perceptron s'écrit alors:

$$y = \sigma(\mathbf{w}^T \mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{w}^T \mathbf{x} \geq 0 \\ -1, & \text{if } \mathbf{w}^T \mathbf{x} < 0. \end{cases}$$



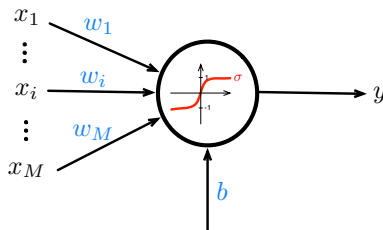
PERCEPTRON

- ▶ Par abus de langage, posons:

$$\mathbf{x} := (1, x_1, \dots, x_M) \text{ et } \mathbf{w} := (b, w_1, \dots, w_M).$$

- ▶ La dynamique du perceptron s'écrit alors:

$$y = \sigma(\mathbf{w}^T \mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{w}^T \mathbf{x} \geq 0 \\ -1, & \text{if } \mathbf{w}^T \mathbf{x} < 0. \end{cases}$$



REMARQUE

- ▶ Soient les paramètres $\mathbf{w} = (w_0, w_1, \dots, w_M) \in \mathbb{R}^{M+1}$, où $w_0 = b$.
- ▶ Le vecteur \mathbf{w} définit un *hyperplan* dans l'espace des inputs \mathbb{R}^M (en non \mathbb{R}^{M+1}) dont l'équation est

$$\mathbf{w}^T \mathbf{x} = 0 \quad \text{i.e.,} \quad \sum_{i=1}^M w_i x_i + b = 0.$$

- ▶ $\mathbf{w}' := (w_1, \dots, w_M)$ est le vecteur normal de cet hyperplan.
- ▶ Par définition, le perceptron de paramètres \mathbf{w} classe les points de \mathbb{R}^M de part et d'autre de cet hyperplan.

REMARQUE

- ▶ Soient les paramètres $\mathbf{w} = (w_0, w_1, \dots, w_M) \in \mathbb{R}^{M+1}$, où $w_0 = b$.
- ▶ Le vecteur \mathbf{w} définit un *hyperplan* dans l'espace des inputs \mathbb{R}^M (en non \mathbb{R}^{M+1}) dont l'équation est

$$\mathbf{w}^T \mathbf{x} = 0 \quad \text{i.e.,} \quad \sum_{i=1}^M w_i x_i + b = 0.$$

- ▶ $\mathbf{w}' := (w_1, \dots, w_M)$ est le vecteur normal de cet hyperplan.
- ▶ Par définition, le perceptron de paramètres \mathbf{w} classe les points de \mathbb{R}^M de part et d'autre de cet hyperplan.

REMARQUE

- ▶ Soient les paramètres $\mathbf{w} = (w_0, w_1, \dots, w_M) \in \mathbb{R}^{M+1}$, où $w_0 = b$.
- ▶ Le vecteur \mathbf{w} définit un *hyperplan* dans l'espace des inputs \mathbb{R}^M (en non \mathbb{R}^{M+1}) dont l'équation est

$$\mathbf{w}^T \mathbf{x} = 0 \quad \text{i.e.,} \quad \sum_{i=1}^M w_i x_i + b = 0.$$

- ▶ $\mathbf{w}' := (w_1, \dots, w_M)$ est le vecteur normal de cet hyperplan.
- ▶ Par définition, le perceptron de paramètres \mathbf{w} classe les points de \mathbb{R}^M de part et d'autre de cet hyperplan.

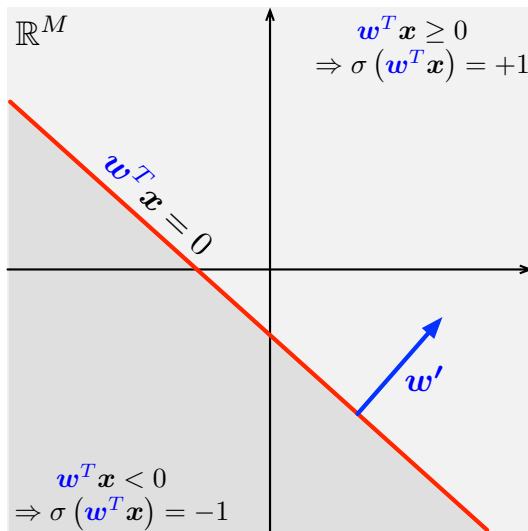
REMARQUE

- ▶ Soient les paramètres $\mathbf{w} = (w_0, w_1, \dots, w_M) \in \mathbb{R}^{M+1}$,
où $w_0 = b$.
- ▶ Le vecteur \mathbf{w} définit un *hyperplan* dans l'espace des inputs \mathbb{R}^M
(en non \mathbb{R}^{M+1}) dont l'équation est

$$\mathbf{w}^T \mathbf{x} = 0 \quad \text{i.e.,} \quad \sum_{i=1}^M w_i x_i + b = 0.$$

- ▶ $\mathbf{w}' := (w_1, \dots, w_M)$ est le vecteur normal de cet hyperplan.
- ▶ Par définition, le perceptron de paramètres \mathbf{w} classe les points de \mathbb{R}^M de part et d'autre de cet hyperplan.

REMARQUE



ENTRAÎNEMENT (TRAINING)

- Soit un train set

$$\mathcal{S} = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \dots, K\}.$$

- L'*entraînement* d'un perceptron (*training*) consiste à déterminer (s'ils existent) des poids $\hat{\mathbf{w}}$ tels que tous les points du train set soient bien classifiés, i.e.:

$$\text{Si } y_k = +1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = +1$$

$$\text{Si } y_k = -1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = -1$$

- Si les points ne sont pas linéairement séparables, on peut fixer un critère d'arrêt:

$$\frac{1}{K} \sum_{k=0}^K (y_k - \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k)) < \delta.$$

ENTRAÎNEMENT (TRAINING)

- Soit un train set

$$\mathcal{S} = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \dots, K\}.$$

- L'*entraînement* d'un perceptron (*training*) consiste à déterminer (s'ils existent) des poids $\hat{\mathbf{w}}$ tels que tous les points du train set soient bien classifiés, i.e.:

$$\text{Si } y_k = +1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = +1$$

$$\text{Si } y_k = -1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = -1$$

- Si les points ne sont pas linéairement séparables, on peut fixer un critère d'arrêt:

$$\frac{1}{K} \sum_{k=0}^K (y_k - \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k)) < \delta.$$

ENTRAÎNEMENT (TRAINING)

- Soit un train set

$$\mathcal{S} = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \dots, K\}.$$

- L'*entraînement* d'un perceptron (*training*) consiste à déterminer (s'ils existent) des poids $\hat{\mathbf{w}}$ tels que tous les points du train set soient bien classifiés, i.e.:

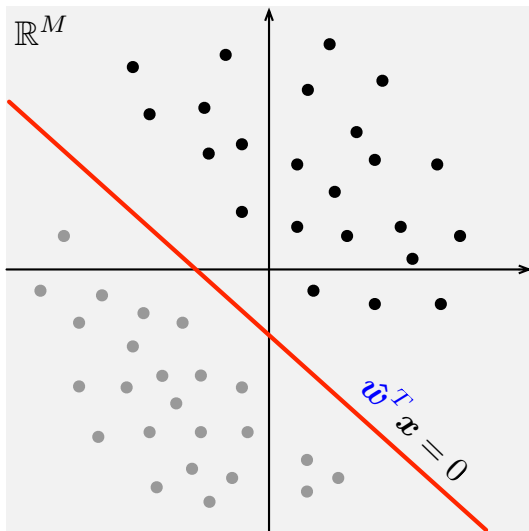
$$\text{Si } y_k = +1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = +1$$

$$\text{Si } y_k = -1, \text{ alors } \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k) = -1$$

- Si les points ne sont pas linéairement séparables, on peut fixer un critère d'arrêt:

$$\frac{1}{K} \sum_{k=0}^K (y_k - \sigma(\hat{\mathbf{w}}^T \mathbf{x}_k)) < \delta.$$

ENTRAÎNEMENT (TRAINING)



ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

```
w := (0, ..., 0, 0) = 0
for e = 1 to nb_epochs do
  for k = 1 to K do
    if  $y_k = -1$  and  $\sigma(w^T x_k) = +1$  then
      w := w -  $x_k$ 
    else if  $y_k = +1$  and  $\sigma(w^T x_k) = -1$  then
      w := w +  $x_k$ 
    end
  end
end
return predictions
```

ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ to nb_epochs do

 for $k = 1$ to K do

 if $y_k = -1$ and $\sigma(\mathbf{w}^T \mathbf{x}_k) = +1$ then

$\mathbf{w} := \mathbf{w} - \mathbf{x}_k$

 else if $y_k = +1$ and $\sigma(\mathbf{w}^T \mathbf{x}_k) = -1$ then

$\mathbf{w} := \mathbf{w} + \mathbf{x}_k$

 end

 end

end

return predictions

ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ *to* nb_epochs **do**

for $k = 1$ *to* K **do**

if $y_k = -1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = +1$ **then**

$\mathbf{w} := \mathbf{w} - \mathbf{x}_k$

else if $y_k = +1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = -1$ **then**

$\mathbf{w} := \mathbf{w} + \mathbf{x}_k$

end

end

end

return predictions

ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ *to* nb_epochs **do**

for $k = 1$ *to* K **do**

if $y_k = -1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = +1$ **then**

$\mathbf{w} := \mathbf{w} - \mathbf{x}_k$

else if $y_k = +1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = -1$ **then**

$\mathbf{w} := \mathbf{w} + \mathbf{x}_k$

end

end

end

return predictions

ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ *to* nb_epochs **do**

for $k = 1$ *to* K **do**

if $y_k = -1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = +1$ **then**

$\mathbf{w} := \mathbf{w} - \mathbf{x}_k$

else if $y_k = +1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = -1$ **then**

$\mathbf{w} := \mathbf{w} + \mathbf{x}_k$

end

end

end

return predictions

ENTRAÎNEMENT (TRAINING)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$ $\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$ **for** $e = 1$ *to* nb_epochs **do** **for** $k = 1$ *to* K **do** **if** $y_k = -1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = +1$ **then** $\mathbf{w} := \mathbf{w} - \mathbf{x}_k$ **else if** $y_k = +1$ *and* $\sigma(\mathbf{w}^T \mathbf{x}_k) = -1$ **then** $\mathbf{w} := \mathbf{w} + \mathbf{x}_k$ **end** **end****end****return** predictions

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

```
 $w := (0, \dots, 0, 0) = \mathbf{0}$ 
for  $e = 1$  to  $nb\_epochs$  do
  for  $k = 1$  to  $K$  do
    if  $y_k \cdot \sigma(w^T x_k) < 0$  then
       $w := w + y_k \cdot x_k$ 
    end
  end
end
return predictions
```

- On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $w := w + \gamma \cdot y_k \cdot x_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

```
for  $e = 1$  to  $nb\_epochs$  do
  for  $k = 1$  to  $K$  do
    if  $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$  then
       $\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$ 
    end
  end
end
return predictions
```

- On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ **to** nb_epochs **do**

for $k = 1$ **to** K **do**

if $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$ **then**

$\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$

end

end

end

return predictions

- ▶ On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ **to** nb_epochs **do**

for $k = 1$ **to** K **do**

if $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$ **then**

$\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$

end

end

end

return predictions

- ▶ On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ **to** nb_epochs **do**

for $k = 1$ **to** K **do**

if $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$ **then**

$\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$

end

end

end

return predictions

- ▶ On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ **to** nb_epochs **do**

for $k = 1$ **to** K **do**

if $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$ **then**

$\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$

end

end

end

return predictions

- On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

ENTRAÎNEMENT (TRAINING)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\mathbf{x}_i, y_i) : i = 1, \dots, K\}$

$\mathbf{w} := (0, \dots, 0, 0) = \mathbf{0}$

for $e = 1$ **to** nb_epochs **do**

for $k = 1$ **to** K **do**

if $y_k \cdot \sigma(\mathbf{w}^T \mathbf{x}_k) < 0$ **then**

$\mathbf{w} := \mathbf{w} + y_k \cdot \mathbf{x}_k$

end

end

end

return predictions

- ▶ On peut utiliser un *learning rate* $\gamma > 0$, ce qui modifie la règle de mise à jour en $\mathbf{w} := \mathbf{w} + \gamma \cdot y_k \cdot \mathbf{x}_k$

TRAINING ALGORITHM

```
def train_perceptron(x, y, nb_epochs_max):  
    w = torch.zeros(x.shape[1])           # initial weights (size M)  
    for e in range(nb_epochs_max):        # iterate over epochs  
        nb_changes = 0  
        for i in range(x.shape[0]):       # iterate over train set  
            if w.dot(x[i]) * y[i] <= 0:    # x_i misclassified  
                w = w + (y[i] * x[i, :])    # update weights  
                nb_changes = nb_changes + 1  
        if nb_changes == 0:  
            break  
    return w
```


ILLUSTRATION GÉOMÉTRIQUE

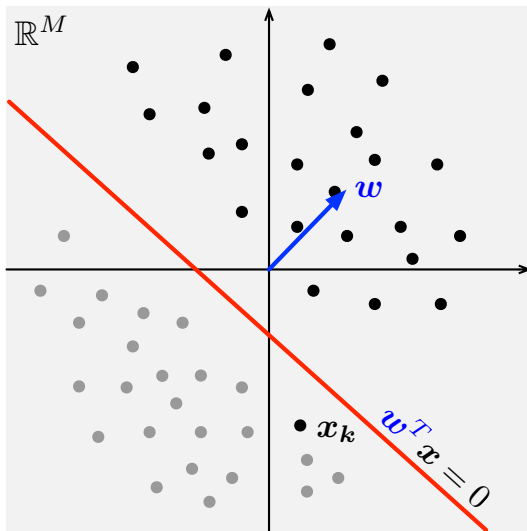


ILLUSTRATION GÉOMÉTRIQUE

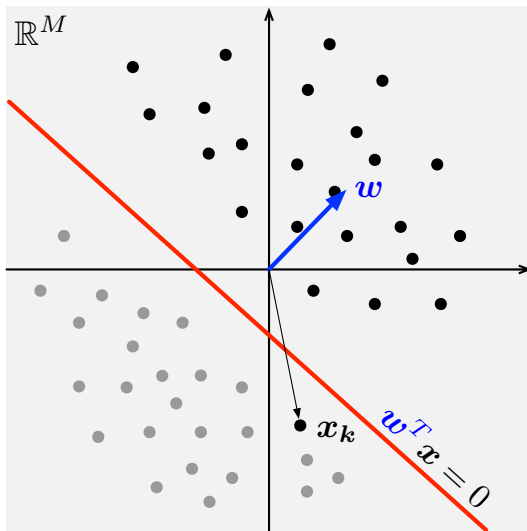


ILLUSTRATION GÉOMÉTRIQUE

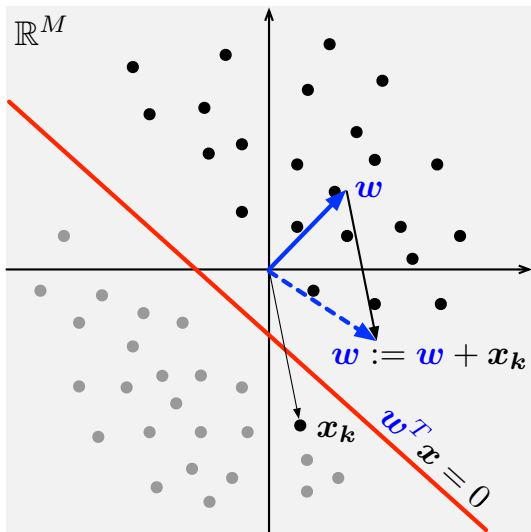


ILLUSTRATION GÉOMÉTRIQUE

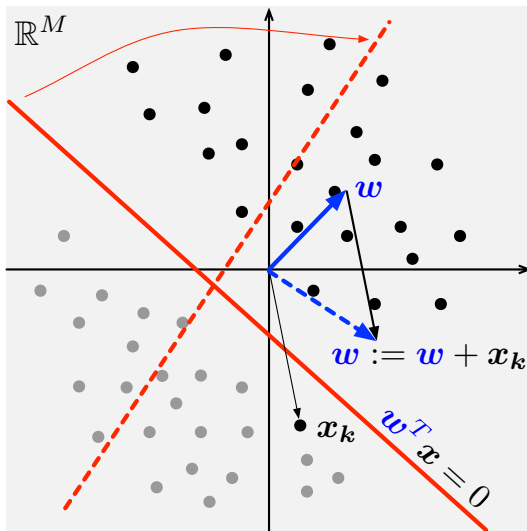


ILLUSTRATION GÉOMÉTRIQUE

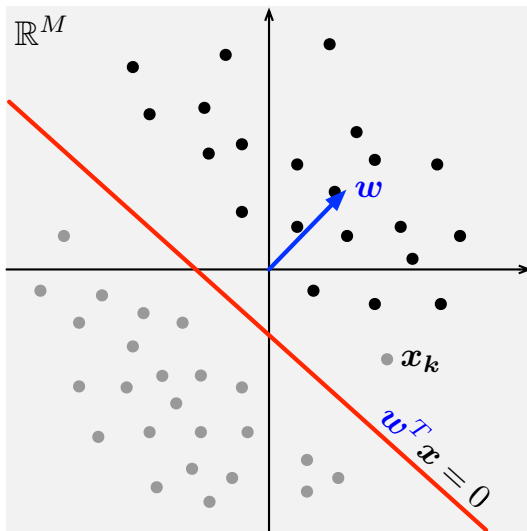


ILLUSTRATION GÉOMÉTRIQUE

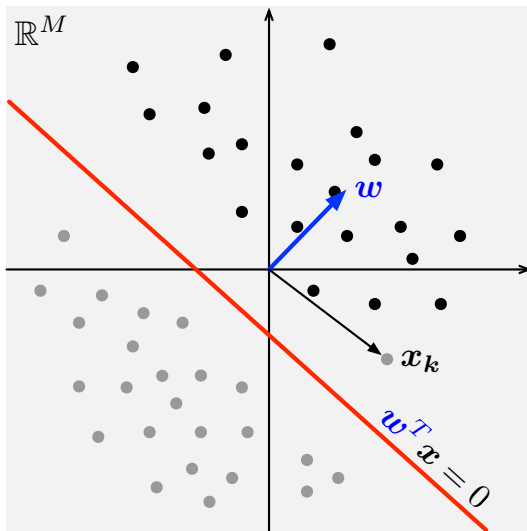


ILLUSTRATION GÉOMÉTRIQUE

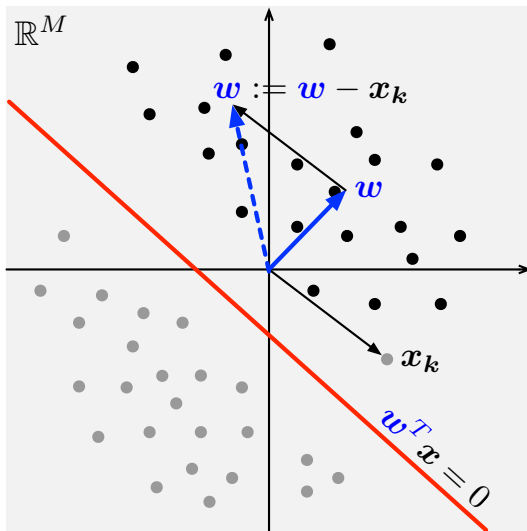
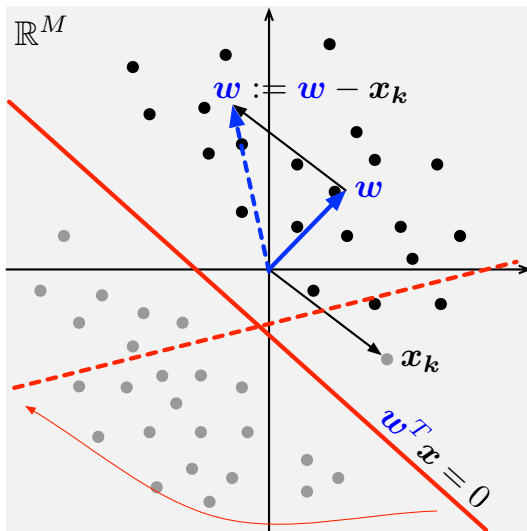
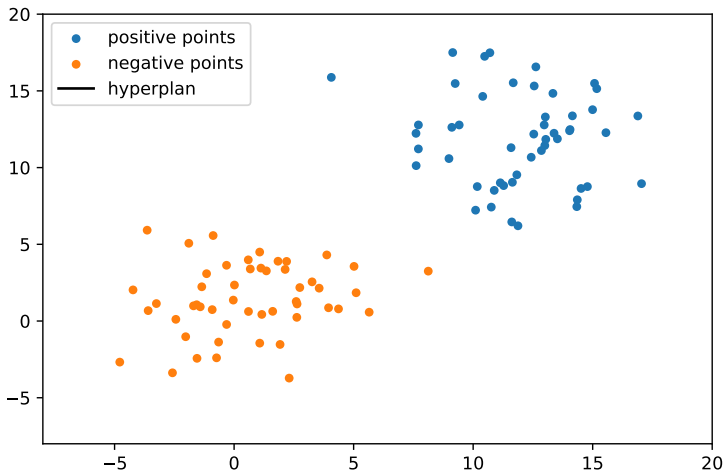


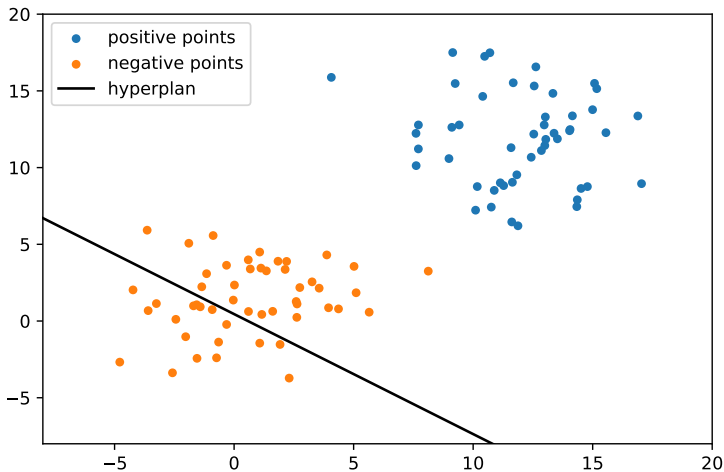
ILLUSTRATION GÉOMÉTRIQUE



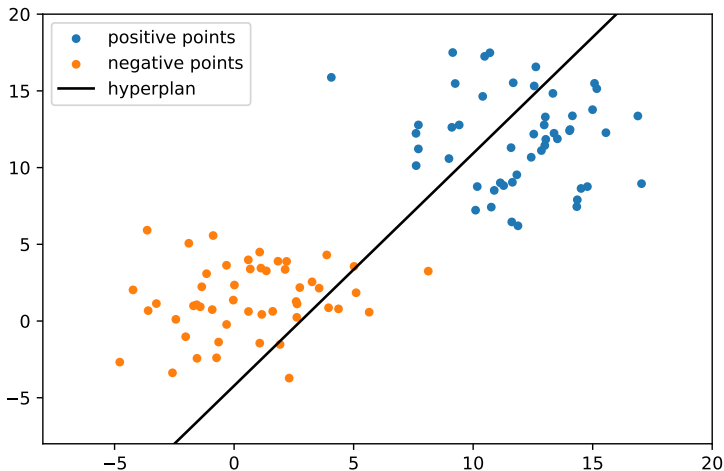
ENTRAÎNEMENT (TRAINING)



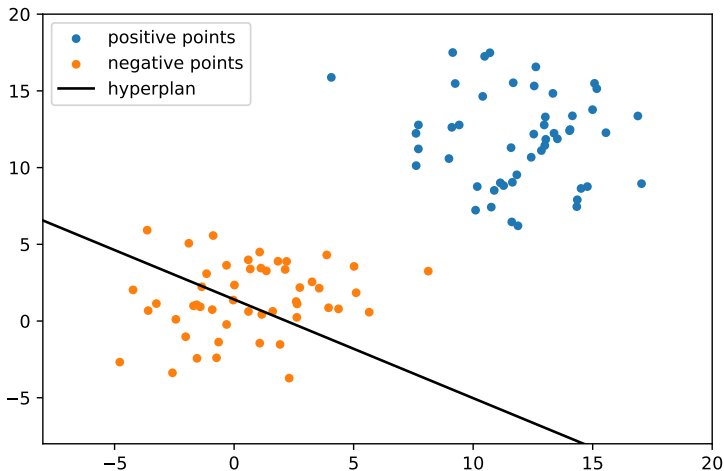
ENTRAÎNEMENT (TRAINING)



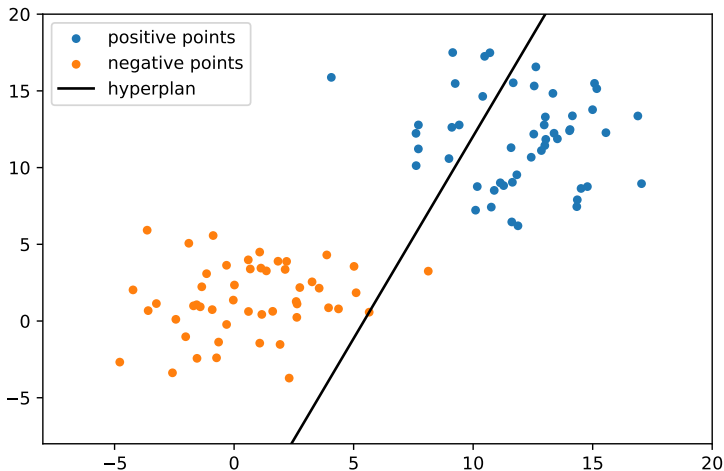
ENTRAÎNEMENT (TRAINING)



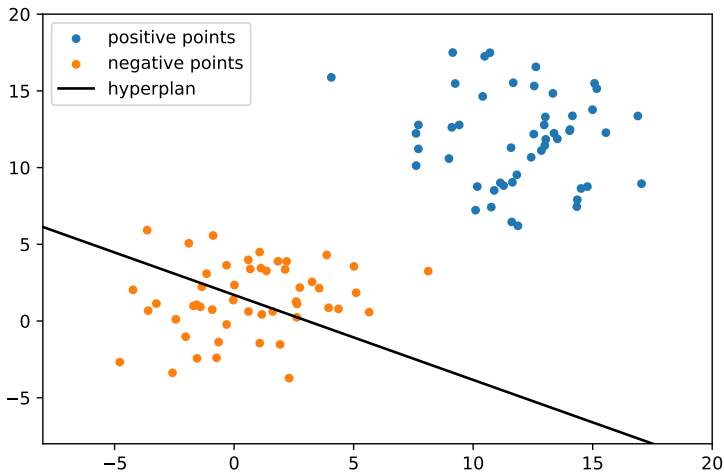
ENTRAÎNEMENT (TRAINING)



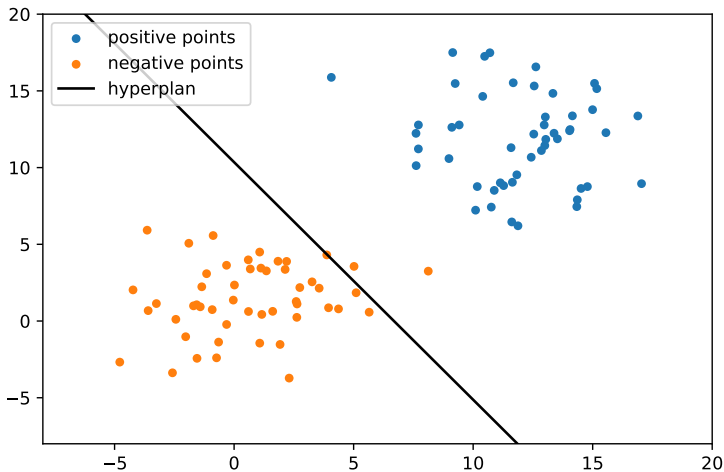
ENTRAÎNEMENT (TRAINING)



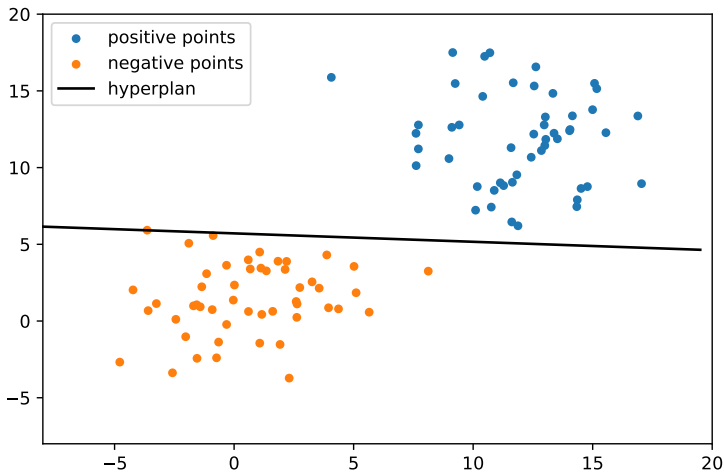
ENTRAÎNEMENT (TRAINING)



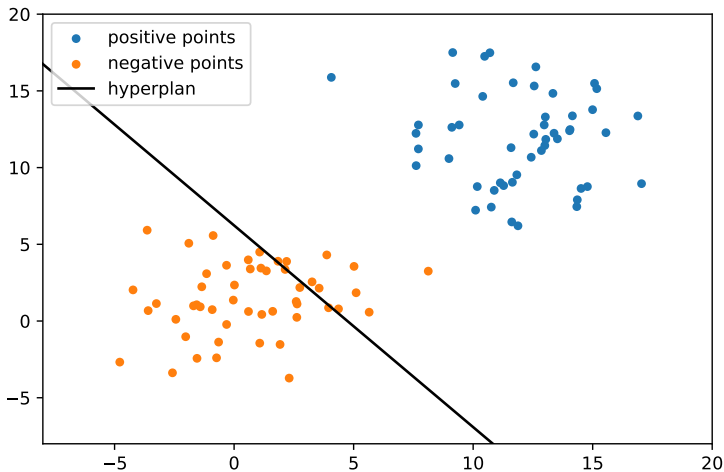
ENTRAÎNEMENT (TRAINING)



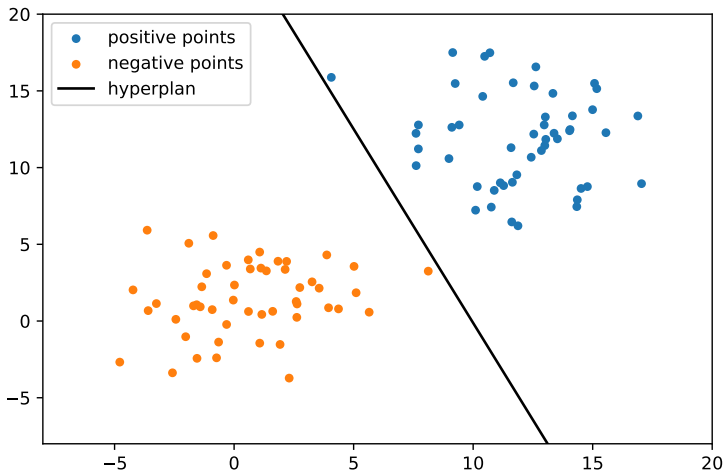
ENTRAÎNEMENT (TRAINING)



ENTRAÎNEMENT (TRAINING)



ENTRAÎNEMENT (TRAINING)



CONVERGENCE

- ▶ Si les points s'inscrivent dans une sphère et satisfont une certaine condition de séparabilité, alors l'algorithme converge.

THEOREM

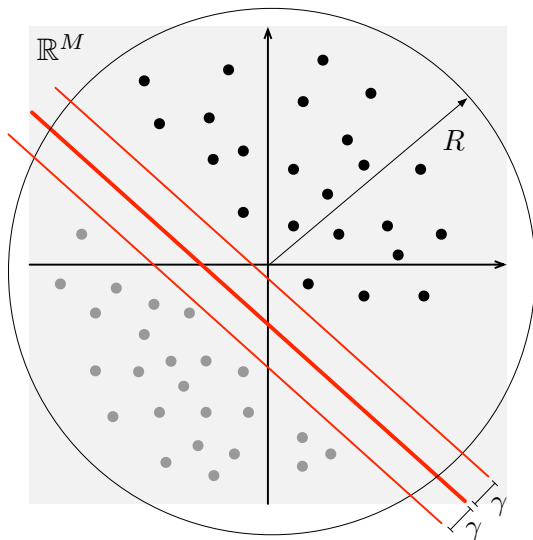
Soit un train set $\mathcal{S} = \{(\mathbf{x}_k, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \dots, K\}$.
Supposons que:

- ▶ Il existe $R > 0$ tel que $\|\mathbf{x}_k\| \leq R$, pour tous $k = 1, \dots, K$;
- ▶ Il existe $\hat{\mathbf{w}} \in \mathbb{R}^{M+1}$ et $\gamma > 0$ tels que $\|\hat{\mathbf{w}}\| = 1$ et

$$y_k \cdot (\hat{\mathbf{w}}^T \mathbf{x}_k) \geq \gamma, \text{ pour tous } k = 1, \dots, K.$$

Alors l'algorithme converge en au plus R^2/γ^2 updates.

CONVERGENCE



BIBLIOGRAPHIE



Fleuret, F. (2022).
Deep Learning Course.



Rosenblatt, F. (1957).
The perceptron: A perceiving and recognizing automaton.
Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York.



Rosenblatt, F. (1958).
The perceptron: A probabilistic model for information storage and organization in
the brain.
Psychological Review, 65(6):386–408.