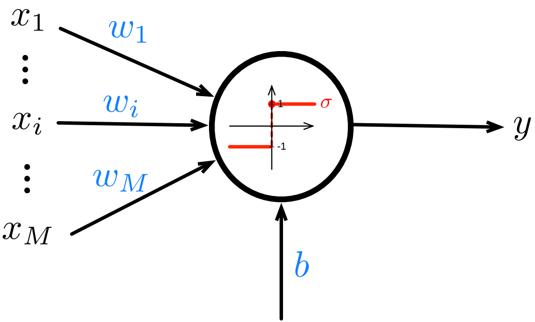
Le perceptron

Jérémie Cabessa

Laboratoire DAVID, UVSQ

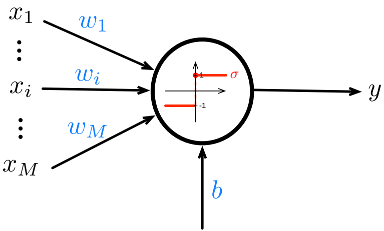
Perceptron

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



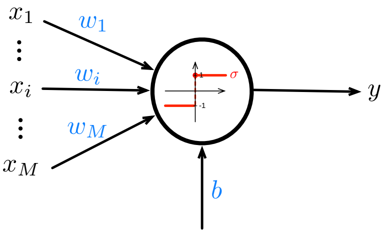
Perceptron

- $\boldsymbol{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- $\boldsymbol{w} = (w_1, \ldots, w_M) \in \mathbb{R}^M$ 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*.



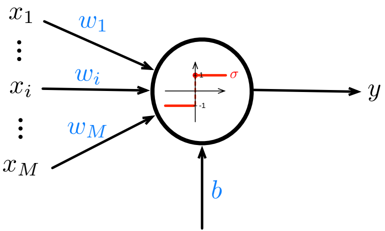
Perceptron

- $\boldsymbol{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- $\boldsymbol{w} = (w_1, \ldots, w_M) \in \mathbb{R}^M$ 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*.



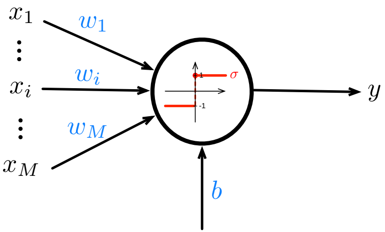
Perceptron

- $\boldsymbol{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- $\boldsymbol{w} = (w_1, \ldots, w_M) \in \mathbb{R}^M$ 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

- $\boldsymbol{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$ sont les *inputs*.
- $\boldsymbol{w} = (w_1, \ldots, w_M) \in \mathbb{R}^M$ 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*.

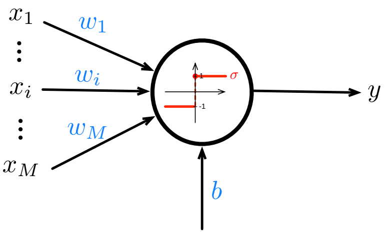


Perceptron

- La dynamique du perceptron est la suivante:

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

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



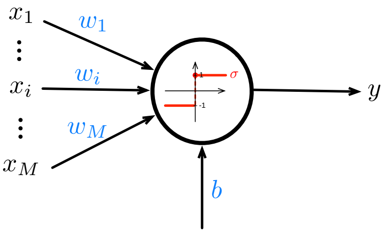
Perceptron

- Par abus de langage, posons:

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

- La dynamique du perceptron s'écrit alors:

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



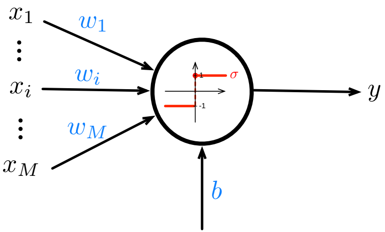
Perceptron

- Par abus de langage, posons:

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

- La dynamique du perceptron s'écrit alors:

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



Remarque

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

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

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

Remarque

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

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

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

Remarque

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

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

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

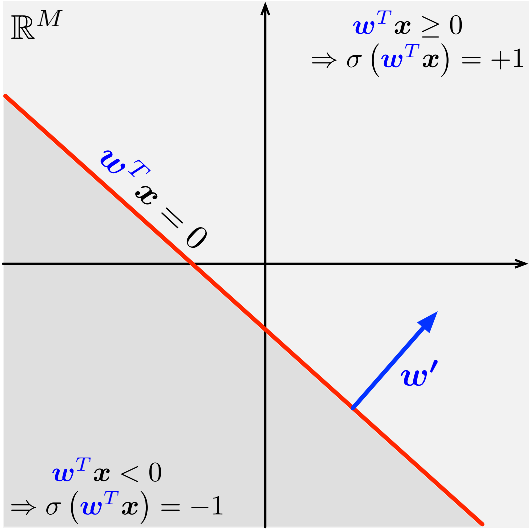
Remarque

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

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

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

Remarque


$\mathbb{R}^M$
$\boldsymbol{w}^T\boldsymbol{x} \geq 0$
$\Rightarrow \sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) = +1$
$\boldsymbol{w}^T\boldsymbol{x} = 0$
$\boldsymbol{w}'$
$\boldsymbol{w}^T\boldsymbol{x} < 0$
$\Rightarrow \sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) = -1$

Remarque

- Si on omet la fonction d'activation σ (ou qu'on prend σ comme étant l'identité), alors la dynamique du perceptron devient

$$y = \boldsymbol{w}^T \boldsymbol{x} + b$$

ce qui correspond exactement à une *régression linéaire*.

Entraînement (training)

- Soit un train set

$$\mathcal{S} = \left\{(\boldsymbol{x_k}, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \ldots, K\right\}.$$

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

$$\text{Si } y_k = +1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = +1$$
$$\text{Si } y_k = -1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = -1$$

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

$$\frac{1}{K}\sum_{k=0}^{K}\left(y_k - \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right)\right) < \delta.$$

Entraînement (training)

- Soit un train set

$$\mathcal{S} = \left\{(\boldsymbol{x_k}, y_k) \in \mathbb{R}^M \times \{-1,+1\} : k = 1, \dots, K\right\}.$$

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

$$\text{Si } y_k = +1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = +1$$
$$\text{Si } y_k = -1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = -1$$

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

$$\frac{1}{K}\sum_{k=0}^{K}\left(y_k - \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right)\right) < \delta.$$

Entraînement (training)

- Soit un train set

$$\mathcal{S} = \left\{ (\boldsymbol{x_k}, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \dots, K \right\}.$$

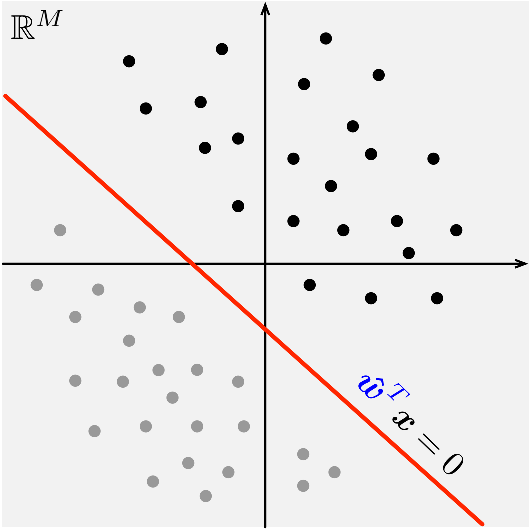
- L'*entraînement* d'un perceptron (*training*) consiste à déterminer (s'ils existent) des poids $\hat{\boldsymbol{w}}$ tels que tous les points du train set soient bien classifiés, i.e.:

$$\text{Si } y_k = +1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = +1$$
$$\text{Si } y_k = -1, \text{ alors } \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right) = -1$$

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

$$\frac{1}{K} \sum_{k=0}^{K} \left(y_k - \sigma\left(\hat{\boldsymbol{w}}^T \boldsymbol{x_k}\right)\right) < \delta.$$

Entraînement (training)



Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

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

for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k = -1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = +1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{x_k}$
    else if $y_k = +1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = -1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{x_k}$
    end
  end
end
return predictions

Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

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

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

Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k = -1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = +1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{x_k}$
    else if $y_k = +1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = -1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{x_k}$
    end
  end
end
return predictions

Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k = -1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = +1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{x_k}$
    else if $y_k = +1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = -1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{x_k}$
    end
  end
end
return predictions

Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k = -1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = +1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{x_k}$
    else if $y_k = +1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = -1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{x_k}$
    end
  end
end
return predictions

Entraînement (training)

Algorithm 1: Training algorithm of the perceptron

Data: dataset = $\{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k = -1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = +1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} - \boldsymbol{x_k}$
    else if $y_k = +1$ *and* $\sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) = -1$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + \boldsymbol{x_k}$
    end
  end
end
return predictions

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

```
w := (0, ..., 0, 0) = 0
for e = 1 to nb_epochs do
    for k = 1 to K do
        if y_k · σ(w^T x_k) < 0 then
            w := w + y_k · 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 $\boldsymbol{w} := \boldsymbol{w} + \gamma \cdot y_k \cdot \boldsymbol{x_k}$

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
    end
  end
end
return predictions

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

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
    end
  end
end
return predictions

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

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
    end
  end
end
return predictions

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

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \mathbf{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
    end
  end
end
return predictions

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

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset = $\{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

$\boldsymbol{w} := (0, \ldots, 0, 0) = \boldsymbol{0}$
for $e = 1$ *to* nb_epochs **do**
  for $k = 1$ *to* K **do**
    if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
      $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
    end
  end
end
return predictions

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

Entraînement (training)

Algorithm 2: Training algorithm of the perceptron (rewriting)

Data: dataset $= \{(\boldsymbol{x_i}, y_i) : i = 1, \ldots, K\}$

```
w := (0, ..., 0, 0) = 0
for e = 1 to nb_epochs do
    for k = 1 to K do
        if y_k · σ(w^T x_k) < 0 then
            w := w + y_k · x_k
        end
    end
end
return predictions
```

$\boldsymbol{w} := (0, \ldots, 0, 0) = \boldsymbol{0}$
for $e = 1$ *to* nb_epochs **do**
 for $k = 1$ *to* K **do**
  if $y_k \cdot \sigma\left(\boldsymbol{w}^T \boldsymbol{x_k}\right) < 0$ **then**
   $\boldsymbol{w} := \boldsymbol{w} + y_k \cdot \boldsymbol{x_k}$
  end
 end
end
return predictions

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

Training algorithm

```python
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

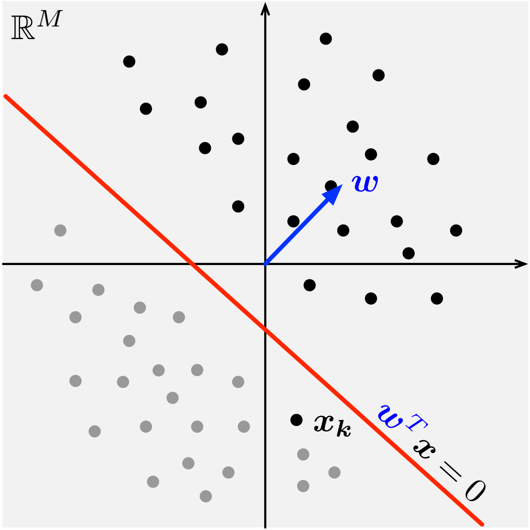


Illustration géométrique

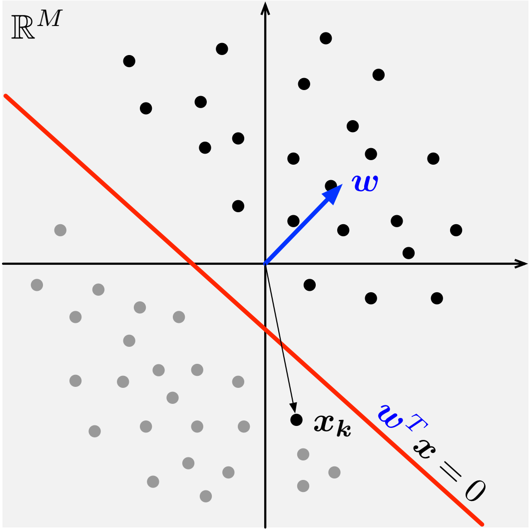


Illustration géométrique

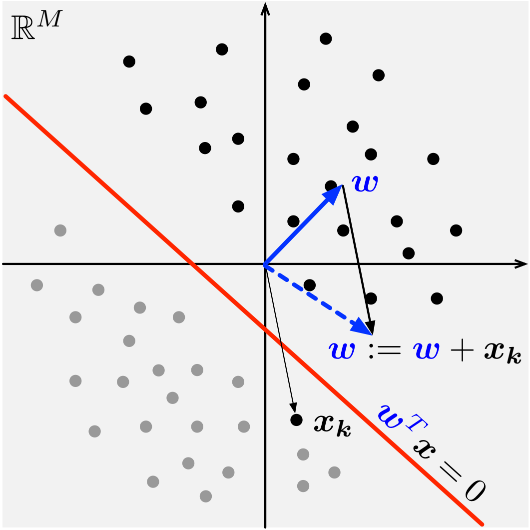


Illustration géométrique

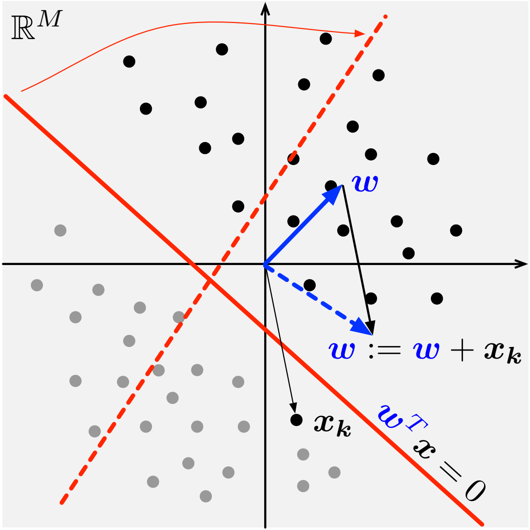


Illustration géométrique

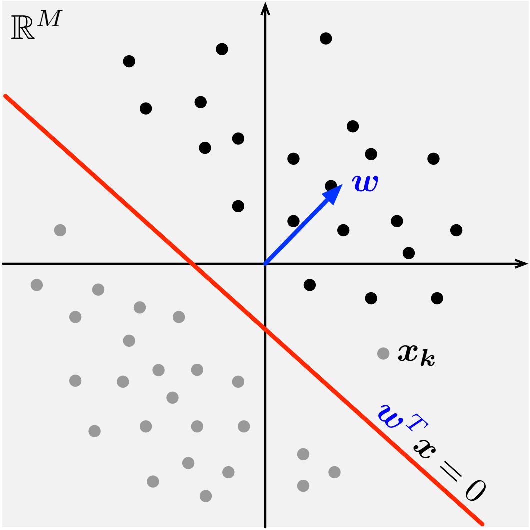


Illustration géométrique

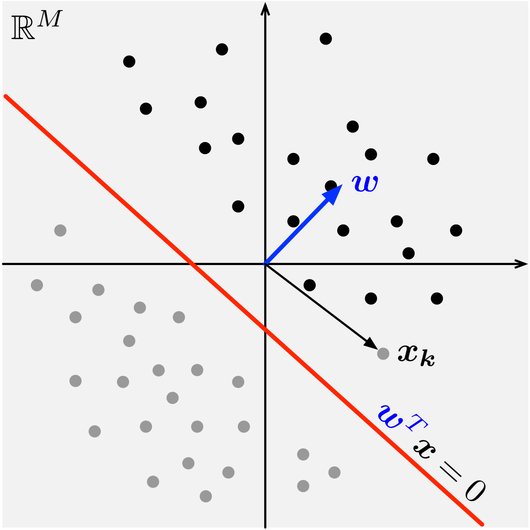


Illustration géométrique

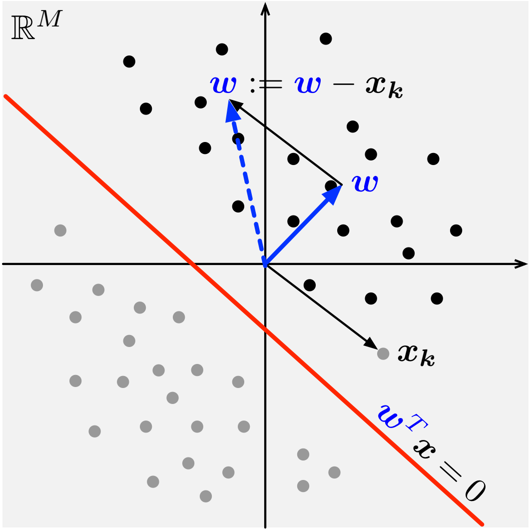
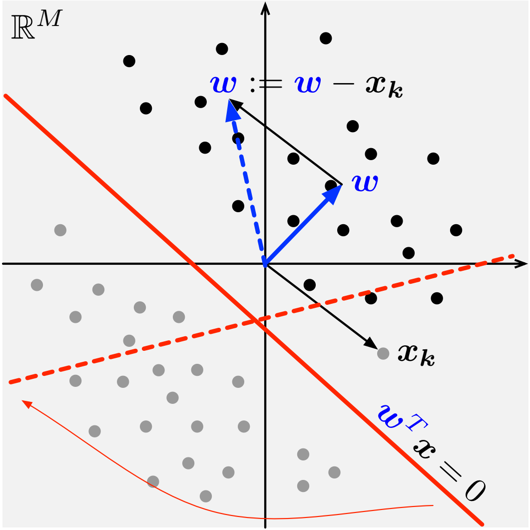
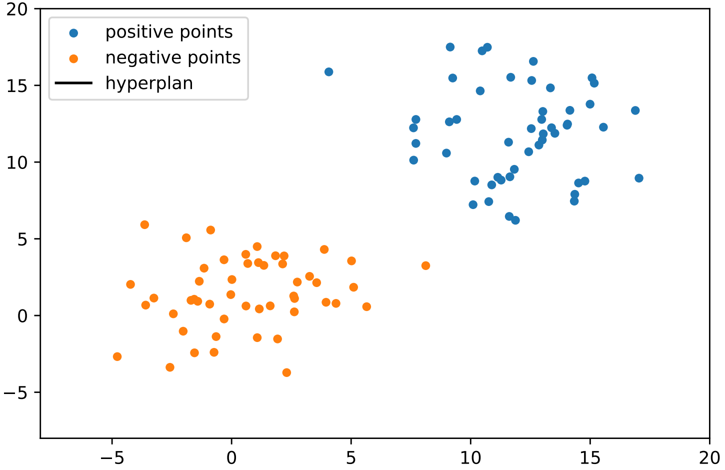


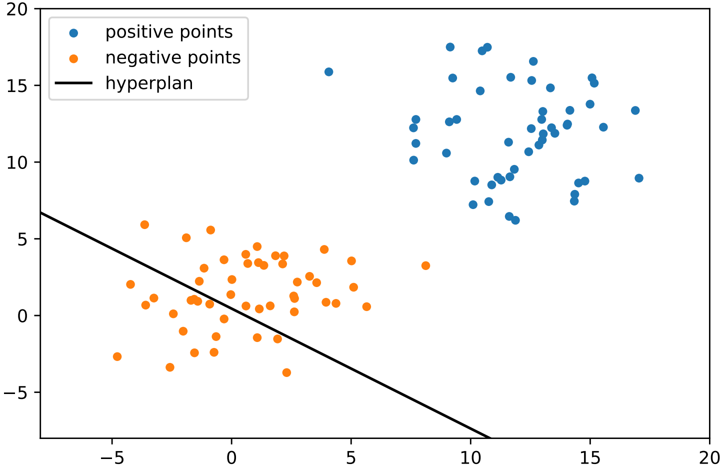
Illustration géométrique



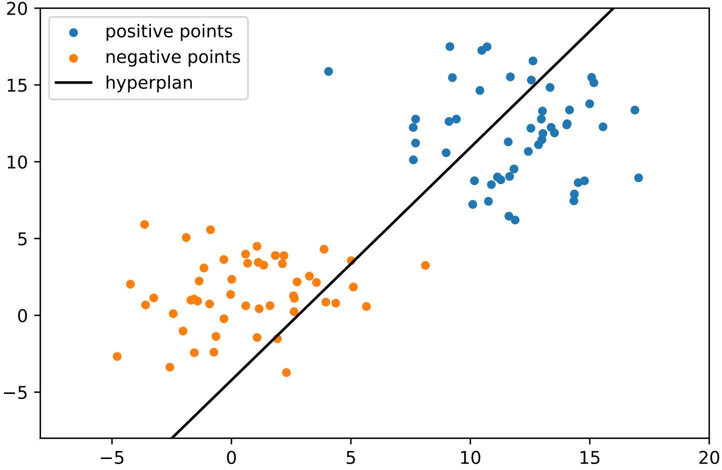
Entraînement (training)



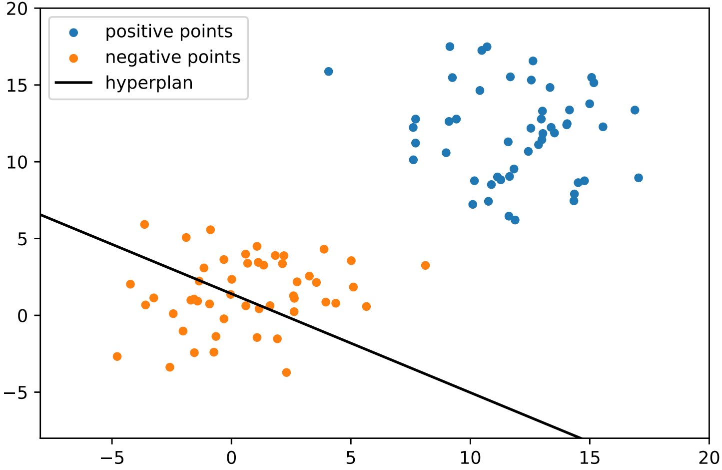
Entraînement (training)



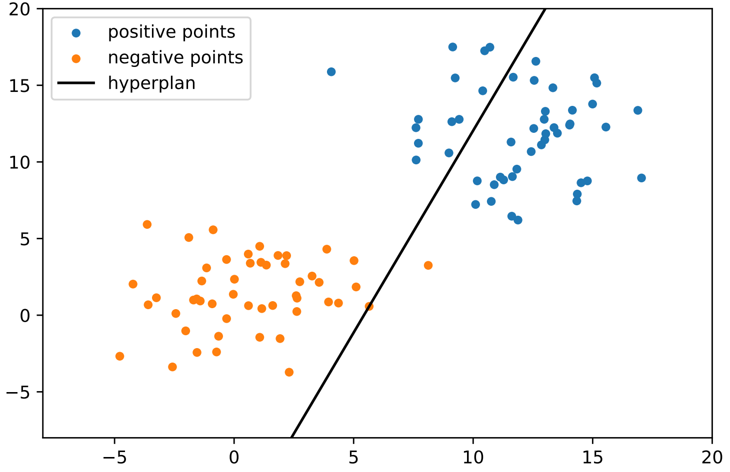
Entraînement (training)



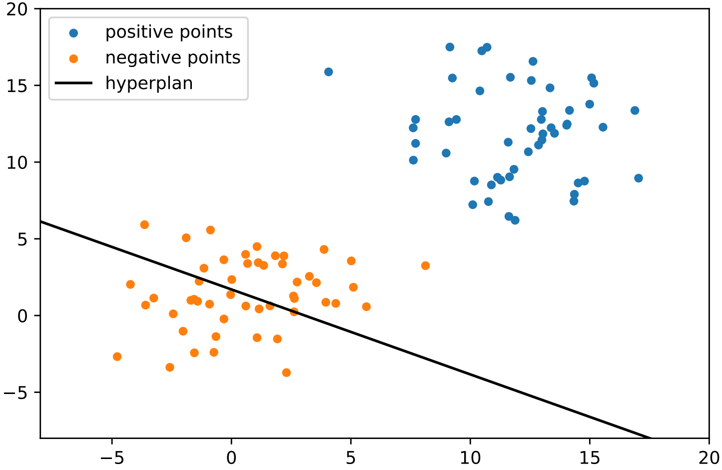
Entraînement (training)



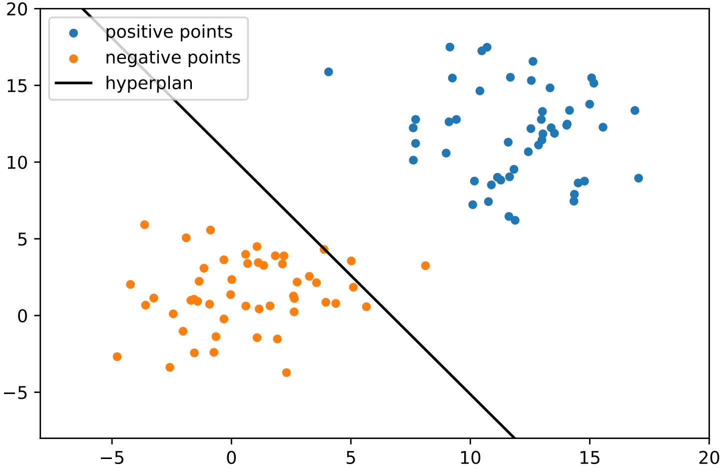
Entraînement (training)



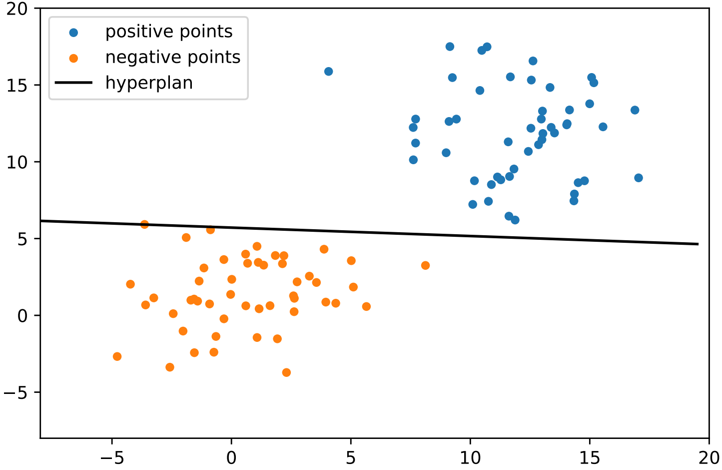
Entraînement (training)



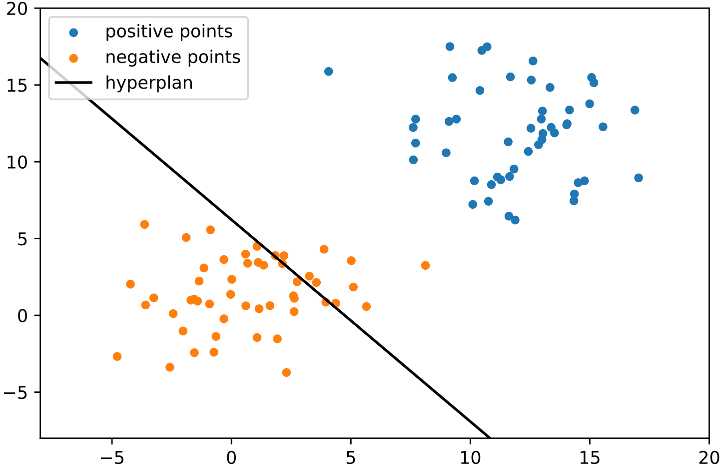
Entraînement (training)



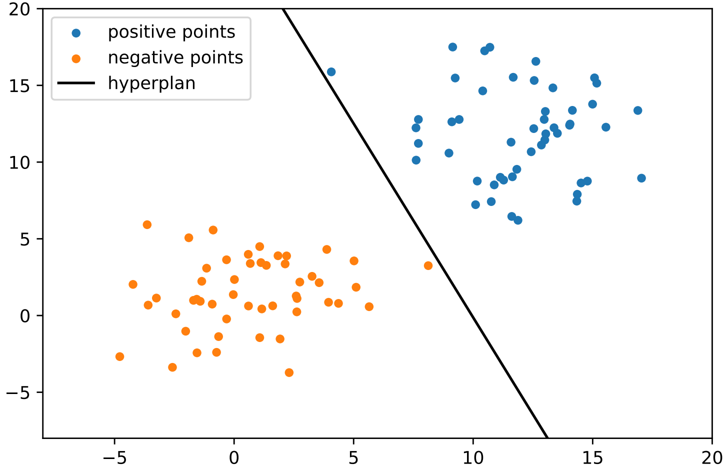
Entraînement (training)



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

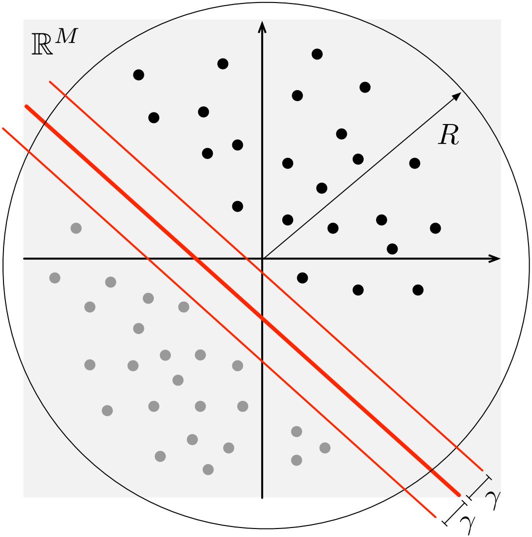
Soit un train set $\mathcal{S} = \left\{(\boldsymbol{x_k}, y_k) \in \mathbb{R}^M \times \{-1, +1\} : k = 1, \cdots K\right\}$*.*
Supposons que:

- *Il existe* $R > 0$ *tel que* $\|\boldsymbol{x_k}\| \leq R$*, pour tous* $k = 1, \cdots K$*;*
- *Il existe* $\hat{\boldsymbol{w}} \in \mathbb{R}^{M+1}$ *et* $\gamma > 0$ *tels que* $\|\hat{\boldsymbol{w}}\| = 1$ *et*

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

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.