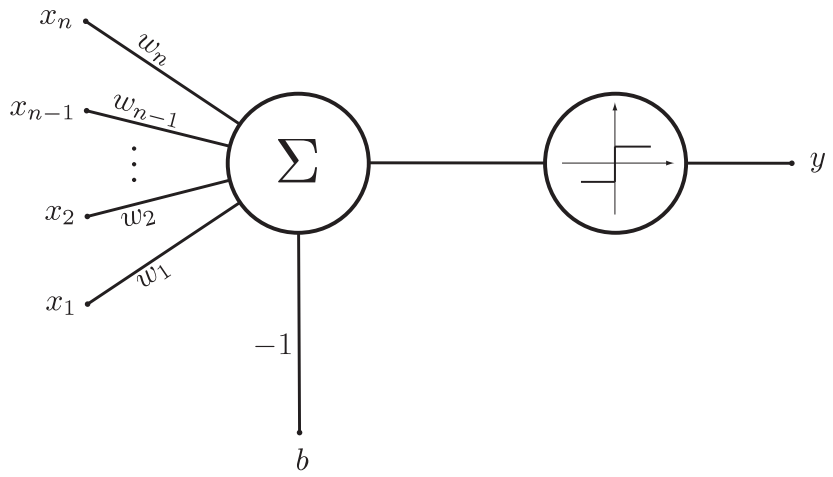
Perceptron Learning

In 1953 Frank Rosenblatt invented another way of calculating a decision surface for a binary classification problem - ***The Perceptron***.

The perceptron is the precursor of our modern artificial neural networks.



It is not difficult to see that a perceptron computes the following decision function

$$\hat{f}(\overline{x}) = y = \text{sign}\left(\left[\sum_{k=1}^{n} w_k x_k\right] + (-1)b\right) = \text{sign}\left(\overline{w} \bullet \overline{x} - b\right).$$

where $\overline{w} = (w_1, w_2, \ldots, w_n)$ and $\overline{x} = (x_1, x_2, \ldots, x_n)$. The free parameters of the perceptron are $\overline{w}$ and b and they need to be estimated using some training set D.

Perceptron Learning

Here $\overline{w}$ and b are called the ***free parameters*** of the perceptron.

Estimating the free parameters using the training set D is called (perceptron) ***learning***. Here D has the usual structure $(\overline{x}, y) \in D$ with $\overline{x} \in \mathbb{R}^n$ and $y \in \{+1, -1\}$.

The ***learning algorithm*** is as follows:

Initialize $\overline{w}$ and b to random values.
Do
  For each $(\overline{x}, y) \in D$ do
    if $\hat{f}(\overline{x}) \neq y$ then
      Update $\overline{w}$ and b
    end if
  end for
Until D is perfectly classified
Return $\overline{w}$ and b

Perceptron Learning

Observations:

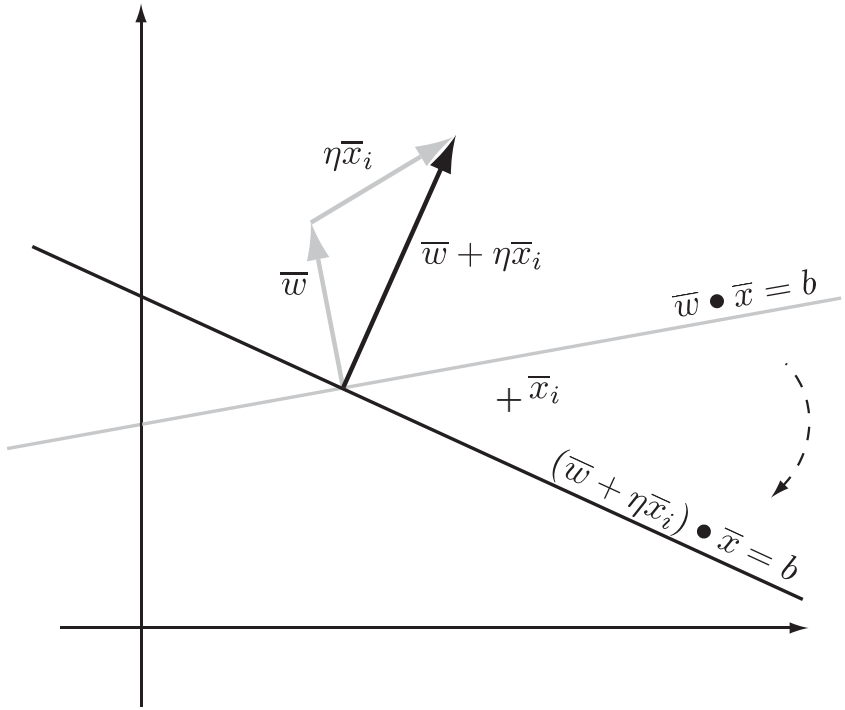
- The inherent assumption is that D is linearly separable, that is, a hyperplane can be found that perfectly separates the two classes. The learning algorithm will not converge if the dataset is not linearly separable.
- The computation stops as soon as no more mistakes are made on classifying the training set D. This might lead to suboptimal decision surfaces. Compare this to our previous learning algorithm where we made sure that the decision surface is placed half way between the to means of the classes, respectively.

Perceptron Learning

let $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$
let $0 < \eta < 1$
$\overline{w} \leftarrow \overline{0}$
$b \leftarrow 0$
$r \leftarrow \max\{|\overline{x}| \mid (\overline{x}, y) \in D\}$
repeat
  for $i = 1$ **to** l
    if $\text{sign}(\overline{w} \bullet \overline{x}_i - b) \neq y_i$ **then**
      $\overline{w} \leftarrow \overline{w} + \eta y_i \overline{x}_i$
      $b \leftarrow b - \eta y_i r^2$
    end if
  end for
until $\text{sign}(\overline{w} \bullet \overline{x}_j - b) = y_j$ with $j = 1, \ldots, l$
return $(\overline{w}, b)$

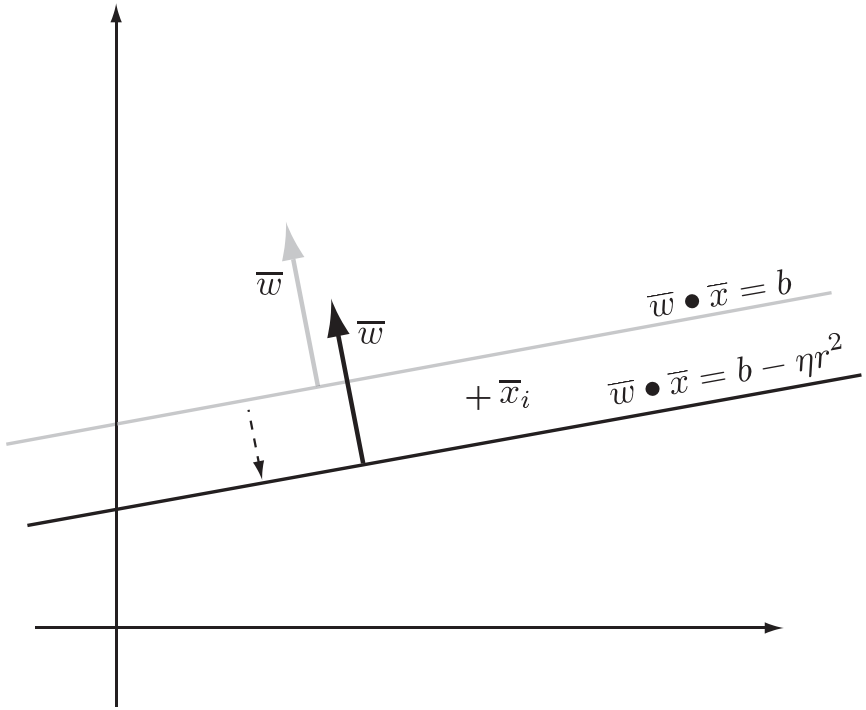
Perceptron Learning

The effect of the perceptron update rule that changes the normal vector of a decision surface $\overline{w} \leftarrow \overline{w} + \eta \overline{x}_i$ using $(\overline{x}_1, +1) \in D$,

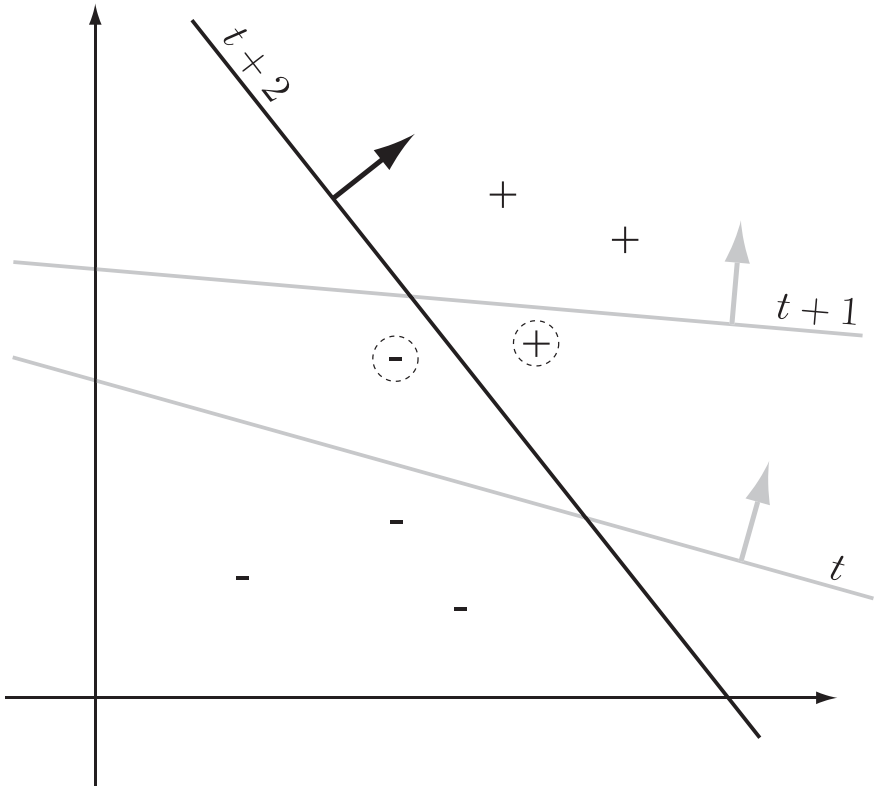


Perceptron Learning

The effect of the perceptron update rule that changes the offset term of a decision surface $b \leftarrow b - \eta r^2$ using $(\overline{x}_1, +1) \in D$,



Perceptron Learning



<demo>

Perceptron Learning

Observations:

- The perceptron learning algorithm is a heuristic search in $\overline{w}$-b-space.
- The search terminates as soon as *some* decision surface has been found.
- There is no guarantee of "optimality".[a]
- However, notice that points far away from the boundary between the classes have very little impact on the solution.

$\Rightarrow$ This means we are kind of stuck, the "simple learning algorithm" constructed an optimal decision surface but was sensitive to outliers; the perceptron is not sensitive to outliers but its learning algorithm provides no guarantees with respect to the quality of the decision surface.

[a] In absence of any other information we consider the optimal decision surface to lie right in the middle between the two classes.