Machine Learning for Intelligent Systems

Lecture 6: Linear Classifiers and Perceptron

Reading: UML 9.1

Instructors: Nika Haghtalab (this time) and Thorsten Joachims

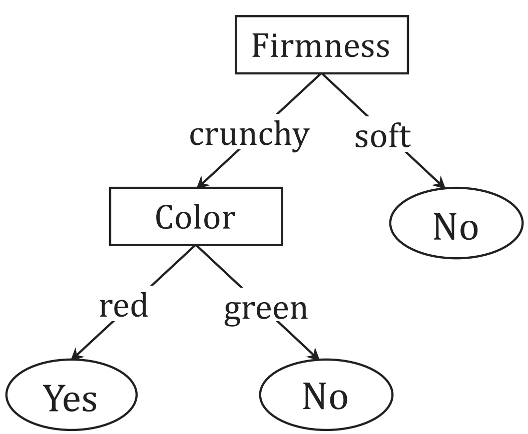
Hypothesis Spaces

$(\text{color} = \text{red}) \wedge (\text{size} = \text{small})$

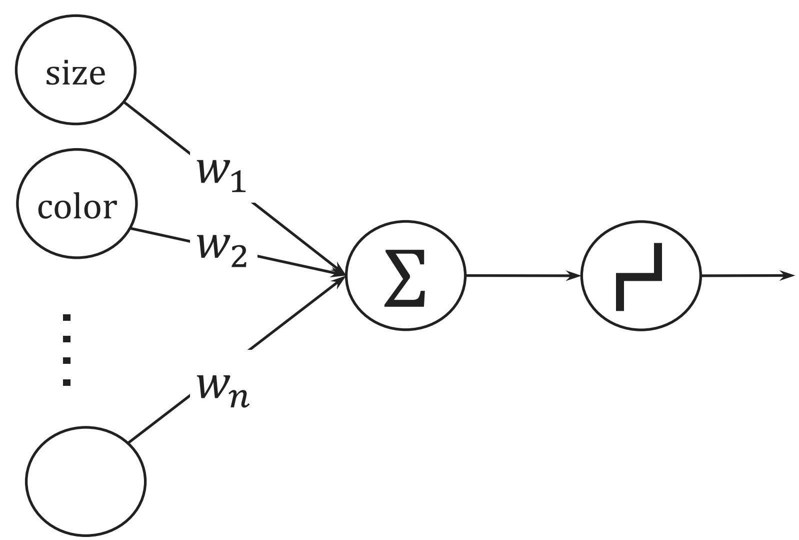
AND of feature-values

$(\text{color} = \text{red}) \vee (\text{size} = \text{small})$

OR of feature-values



Decision Trees



Linear Classifiers

Encoding in Euclidean Space

Represent apples as vectors in $\mathbb{R}^d$.

- Old:

  $X = \{A, B\} \times \{red, green\} \times \{large, medium, small\} \times \{crunchy, soft\}$.
- New:

  → $X \subseteq \mathbb{R}^4$.

  x_1 farm: $A \to 1, B \to -1$

  x_2 color: $red \to 1, green \to -1$

  x_3 size: $large \to 1, medium \to 0, small \to -1$

  x_4 firmness: $crunchy \to 1, \quad soft \to -1.$

  → $Y = \{-1, +1\}$: $Tasty \to +1, Not\ Tasty \to -1$

Reuters Business News text classification:

- 9947 keywords (more accurately, word "stems")
- $X = \{0,1\}^{9947}$, where $x_i = 1$ if the keyword i appears in document.
- $Y = \{-1, +1\}$.

Linear Classifiers

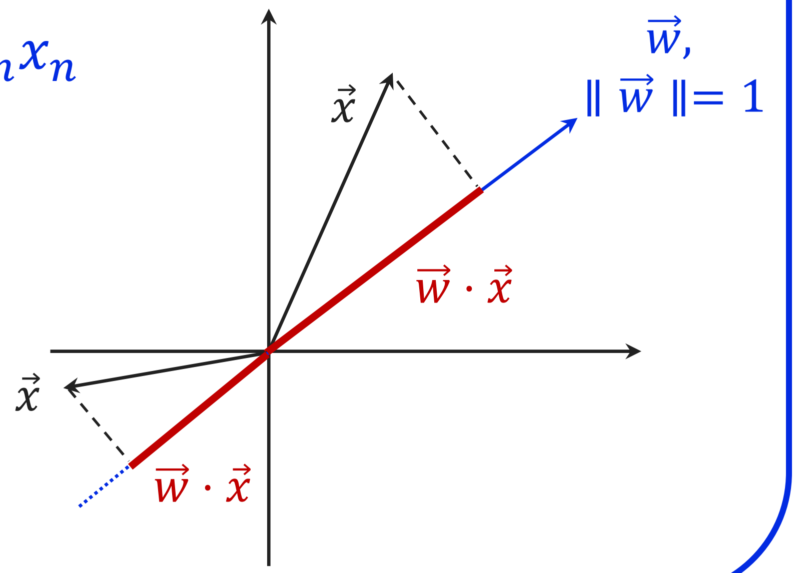
For a vector $\vec{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, the hypothesis $h_{\vec{w},b}: \mathbb{R}^d \to \mathbb{R}$ defined bellow is called a **linear classifier/linear predictor/halfspace,**

$$h_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b) = \begin{cases} +1 & \vec{w} \cdot \vec{x} + b > 0 \\ -1 & \vec{w} \cdot \vec{x} + b \leq 0 \end{cases}$$

Recall: Dot products

For two vectors: $\vec{w} = (w_1, w_2, w_3, \dots, w_n)$ and $\vec{x} = (x_1, x_2, x_3, \dots, x_n)$.

- $\vec{w} \cdot \vec{x} = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n$
- $\vec{w} \cdot \vec{x}$ is the (signed) length of the projection of $\vec{x}$ on unit vector $\vec{w}$.

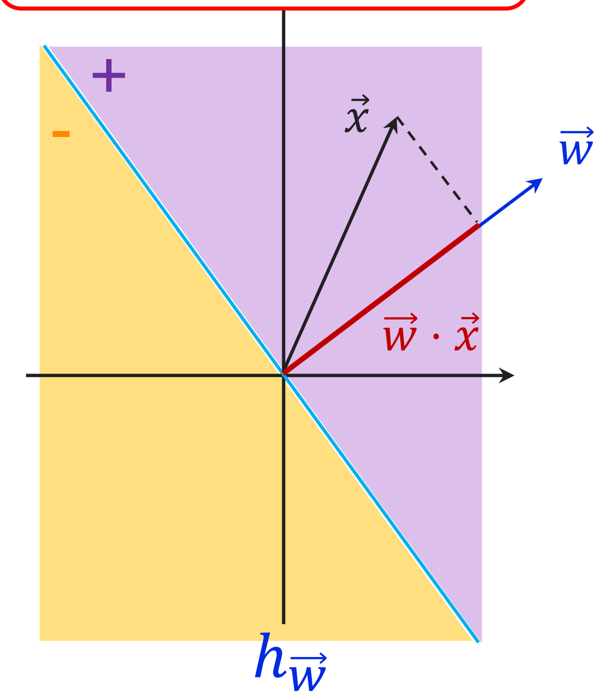


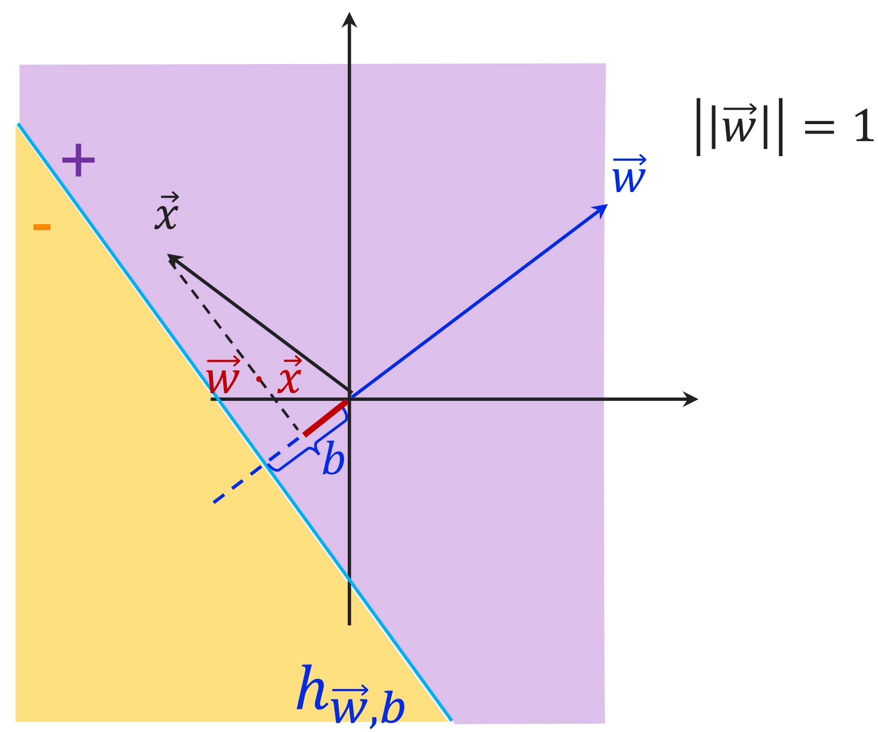
Linear Classifiers

For a vector $\vec{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, the hypothesis $h_{\vec{w},b}: \mathbb{R}^d \to \mathbb{R}$ defined bellow is called a **linear classifier/linear predictor/halfspace**

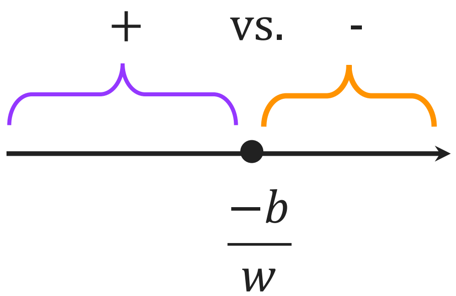
$$h_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b) = \begin{cases} +1 & \vec{w} \cdot \vec{x} + b > 0 \\ -1 & \vec{w} \cdot \vec{x} + b \leq 0 \end{cases}$$

Homogenous linear classifier: $b = 0$

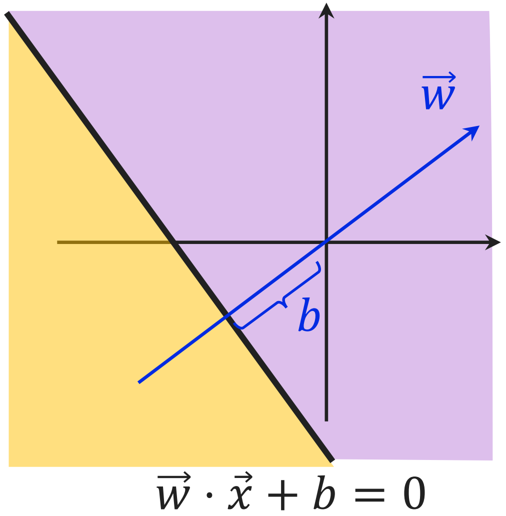




Linear Classifiers in all dimensions



- One-dimension: $h_{w,b}(x) = \mathrm{sign}(wx + b)$
  → Decision boundary: point



- Two-dimension: $h_{\vec{w},b}(x) = \mathrm{sign}(\vec{w} \cdot \vec{x} + b)$
  → Decision boundary: line

- d-dimension: $h_{\vec{w},b}(x) = \mathrm{sign}(\vec{w} \cdot \vec{x} + b)$
  → Decision boundary: hyperplane $\vec{w} \cdot \vec{x} + b = 0$

Representational Power

Assume that x_1, x_2 take are binary values 0, 1. Represent the following using linear thresholds.

- $x_1 \wedge x_2$

- $x_1 \vee x_2$

- $x_1 \oplus x_2$

  → $\oplus$ represent XOR, where $x_1 \oplus x_2 = 1$ when exactly one of x_1 and x_2 is set to 1.

Homogenous vs. Non-homogenous

Any d-dimensional learning problem for **non-homogenous linear classifiers** has a **homogenous** form in (d+1) dimension.

Non-Homogenous $HS^d = \{h_{\vec{\boldsymbol{w}},\boldsymbol{b}} \mid \vec{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$	**Homogenous** $HS^{d+1}_{homogenous} = \{h_{\vec{\boldsymbol{w}}'} \mid \vec{w}' \in \mathbb{R}^{d+1}\}$
$\vec{x}$	$\vec{x}' = (\vec{x}, +1)$
$\vec{w}, b$	$\vec{w}' = (\vec{w}, b)$
$\vec{w} \cdot \vec{x} + b$	$\vec{w}' \cdot \vec{x}' = \vec{w} \cdot \vec{x} + b$

Without loss of generality, focus on **homogenous linear classifiers.**

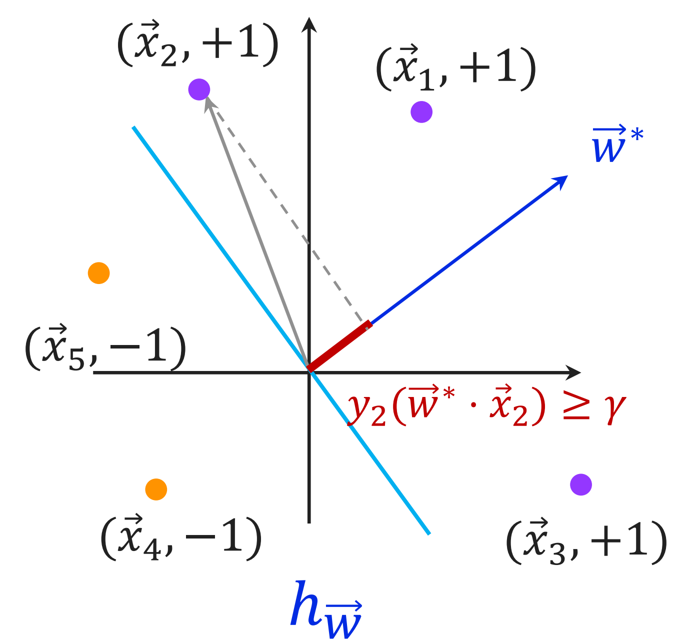
Find a consistent classifier

If there is a homogeneous linear classifier that is consistent with $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_m, y_m)\}$, how can we find it?

Unit vector $\vec{w}^*$ is such that for all $(\vec{x}_i, y_i)$, $\overbrace{y_i(\vec{w}^* \cdot \vec{x}_i) \geq \gamma > 0}^{\text{On the correct side, with wiggle room}}$.

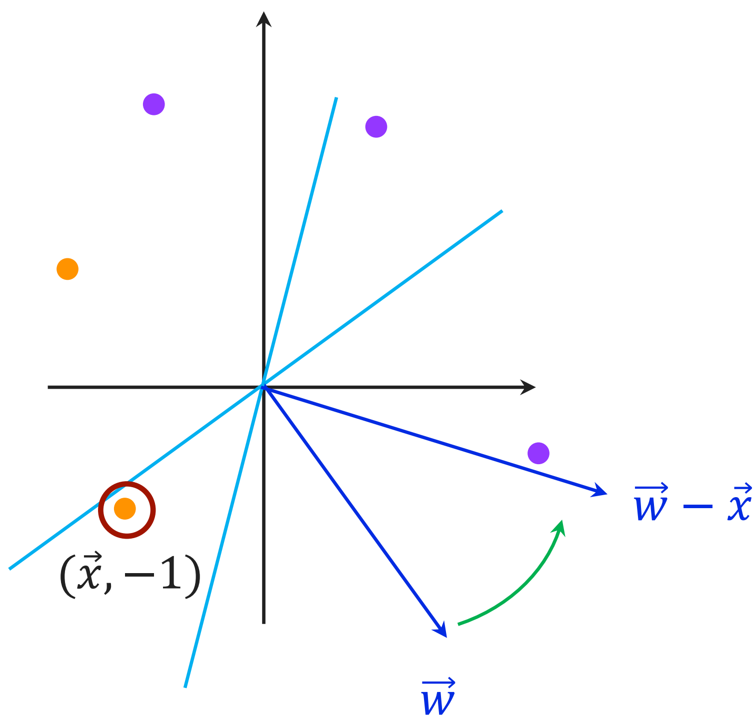
We want to find a $\vec{w}$ such that $y_i(\vec{w} \cdot \vec{x}_i) > 0$ for all $(\vec{x}_i, y_i)$.

Can be done with a linear program

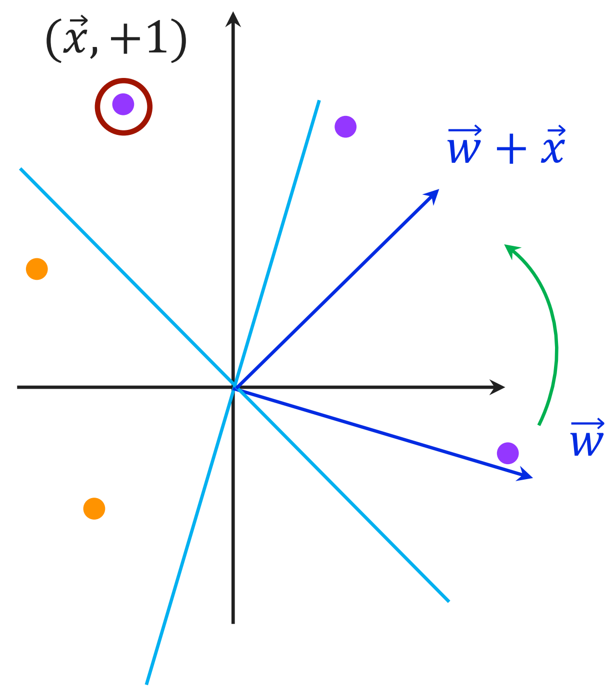


Improving a linear classifier

Start with a guess and improve it.



Move away from negative misclassified points



Move towards positive misclassified points

Perceptron (homogeneous & batch)

Input: Training data set $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_m, y_m)\}$

Initialize $\vec{w}^{(0)} = (0, \dots, 0)$, $t = 0$

While there is $i \in [m]$, such that $\underbrace{y_i\left(\vec{w}^{(t)} \cdot \vec{x}_i\right) \le 0}_{\text{misclassified}}$ then,

- $\vec{w}^{(t+1)} = \vec{w}^{(t)} + y_i\, \vec{x}_i \quad \begin{cases} \vec{w}^{(t)} + \vec{x}_i & \text{for positive instances} \\ \vec{w}^{(t)} - \vec{x}_i & \text{for negative instances} \end{cases}$
- $t \leftarrow t + 1$

End While

Output $\vec{w}^{(t)}$

Frank Rosenblatt
@ Cornell!

Example: Reuters Text Classification