Perceptrons and Optimal Hyperplanes

CS472/CS473 – Fall 2005

Example: Majority-Vote Function

- **Definition: Majority-Vote Function $f_{majority}$**
  - N binary attributes, i.e. $x \in \{0,1\}^N$
  - If more than N/2 attributes in x are true, then $f_{majority}(x)=1$, else $f_{majority}(x)=-1$.
- **How can we represent this function as a decision tree?**
  - Huge and awkward tree!
- **Is there an "easier" representation of $f_{majority}$?**

Example: Spam Filtering

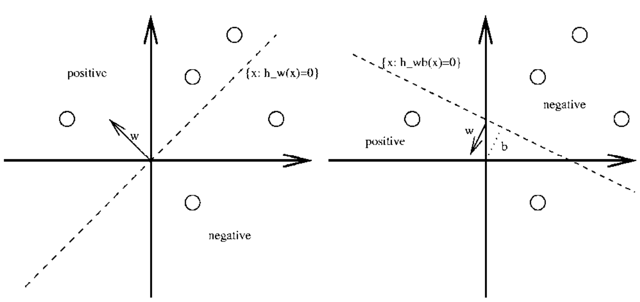
	viagra	learning	the	dating	nigeria	$spam?$
$\vec{x}_1 = ($	1	0	1	0	0 $)$	$y_1 = 1$
$\vec{x}_2 = ($	0	1	1	0	0 $)$	$y_2 = -1$
$\vec{x}_3 = ($	0	0	0	0	1 $)$	$y_3 = 1$

- **Instance Space X:**
  - Feature vector of word occurrences => binary features
  - N features (N typically > 50000)
- **Target Concept c:**
  - Spam (+1) / Ham (-1)
- **Type of function to learn:**
  - Set of Spam words S, Set of Ham words H
  - Classify as Spam (+1), if more Spam words than Ham words in example.

Linear Classification Rules

- **Hypotheses of the form**
  - unbiased: $h_{\vec{w}}(\vec{x}) = \begin{cases} 1 & w_1x_1 + ... + w_Nx_N > 0 \\ -1 & else \end{cases}$
  - biased: $h_{\vec{w},b}(\vec{x}) = \begin{cases} 1 & w_1x_1 + ... + w_Nx_N + b > 0 \\ -1 & else \end{cases}$
  - Parameter vector *w*, scalar *b*
- **Hypothesis space H**
  - $H_{unbiased} = \{h_{\vec{w}} : \vec{w} \in \Re^N\}$
  - $H_{biased} = \{h_{\vec{w},b} : \vec{w} \in \Re^N\ b \in \Re\}$
- **Notation**
  - $w_1x_1 + ... + w_Nx_N = \vec{w} \cdot \vec{x}$ and $sign(a) = \begin{cases} 1 & a > 0 \\ -1 & else \end{cases}$
  - $h_{\vec{w}}(\vec{x}) = sign(\vec{w} \cdot \vec{x})$
  - $h_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$

Geometry of Hyperplane Classifiers



- **Linear Classifiers divide instance space as hyperplane**
- **One side positive, other side negative**

(Batch) Perceptron Algorithm

Input: $D = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$, $\vec{x}_i \in \Re^N$, $y_i \in \{-1, 1\}$, $\eta \in \Re$, $I \in [1, 2, ..]$

Algorithm:

- $\vec{w}_0 = \vec{0}$, $k = 0$
- repeat
  - FOR i=1 TO n
    - IF $y_i(\vec{w}_k \cdot \vec{x}_i) \leq 0$ ### makes mistake
      - $\vec{w}_{k+1} = \vec{w}_k + \eta y_i \vec{x}_i$
      - $k = k + 1$
    - ENDIF
  - ENDFOR
- until I iterations reached

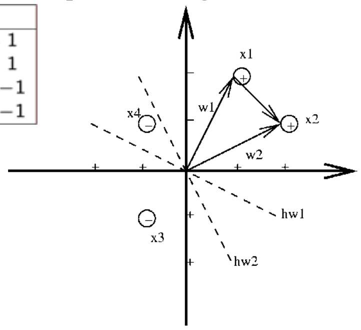
	x_1	x_2	y
$\vec{x}_1 = ($	1	2 $)$	$y_1 = 1$
$\vec{x}_2 = ($	2	1 $)$	$y_2 = 1$
$\vec{x}_3 = ($	-1	-1 $)$	$y_3 = -1$
$\vec{x}_4 = ($	-1	1 $)$	$y_4 = -1$

Example: Perceptron

Training Data:

	x_1	x_2	y
$\vec{x}_1 = ($	1	2)	$y_1 = 1$
$\vec{x}_2 = ($	2	1)	$y_2 = 1$
$\vec{x}_3 = ($	-1	-1)	$y_3 = -1$
$\vec{x}_4 = ($	-1	1)	$y_4 = -1$

Updates to weight vector:



Example: Reuters Text Classification