Linear Classifiers and Perceptron

CS6780 – Advanced Machine Learning
Spring 2019

Thorsten Joachims
Cornell University

Reading: Murphy 8.5.4
Cristianini/Shawe-Taylor Chapter 2-2.1.1

Example: Spam Filtering

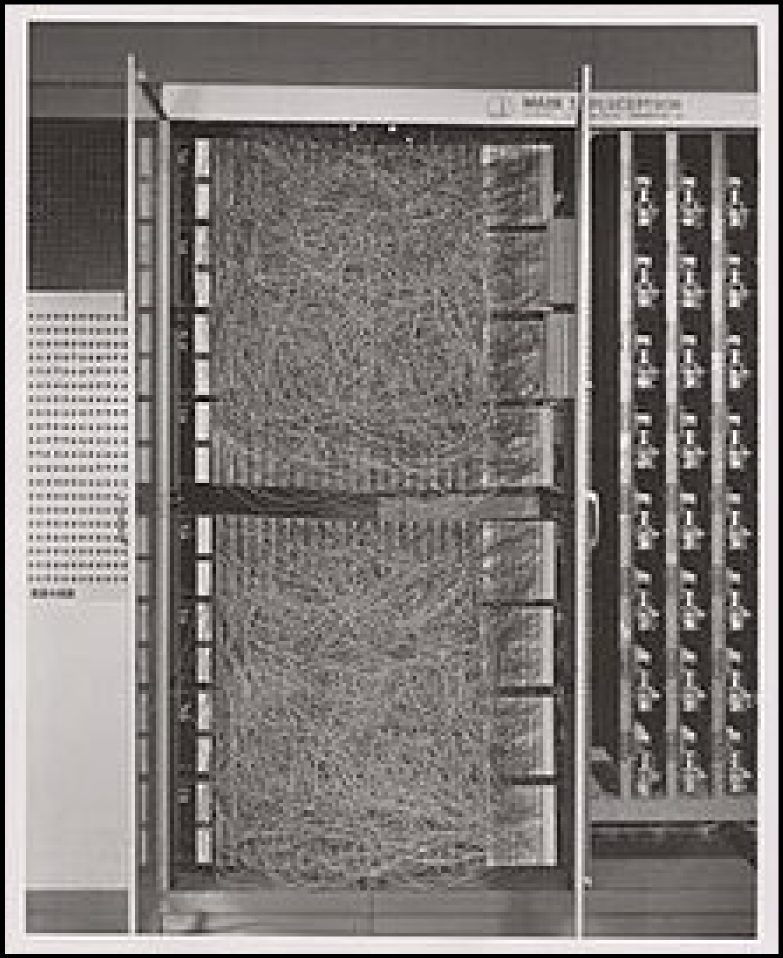
	viagra	learning	the	dating	nigeria	$spam?$
$\vec{x}_1 = ($	1	0	1	0	0 $)$	$y_1 = -1$
$\vec{x}_2 = ($	0	1	1	0	0 $)$	$y_2 = +1$
$\vec{x}_3 = ($	0	0	0	0	1 $)$	$y_3 = -1$

- Instance Space X:
  - Feature vector of word occurrences => binary features
  - N features (N typically > 50000)
- Target Concept c:
  - Spam (-1) / Ham (+1)

Linear Classification Rules

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

Rosenblatt's Perceptron



https://en.wikipedia.org/wiki/Perceptron

(Batch) Perceptron Algorithm

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

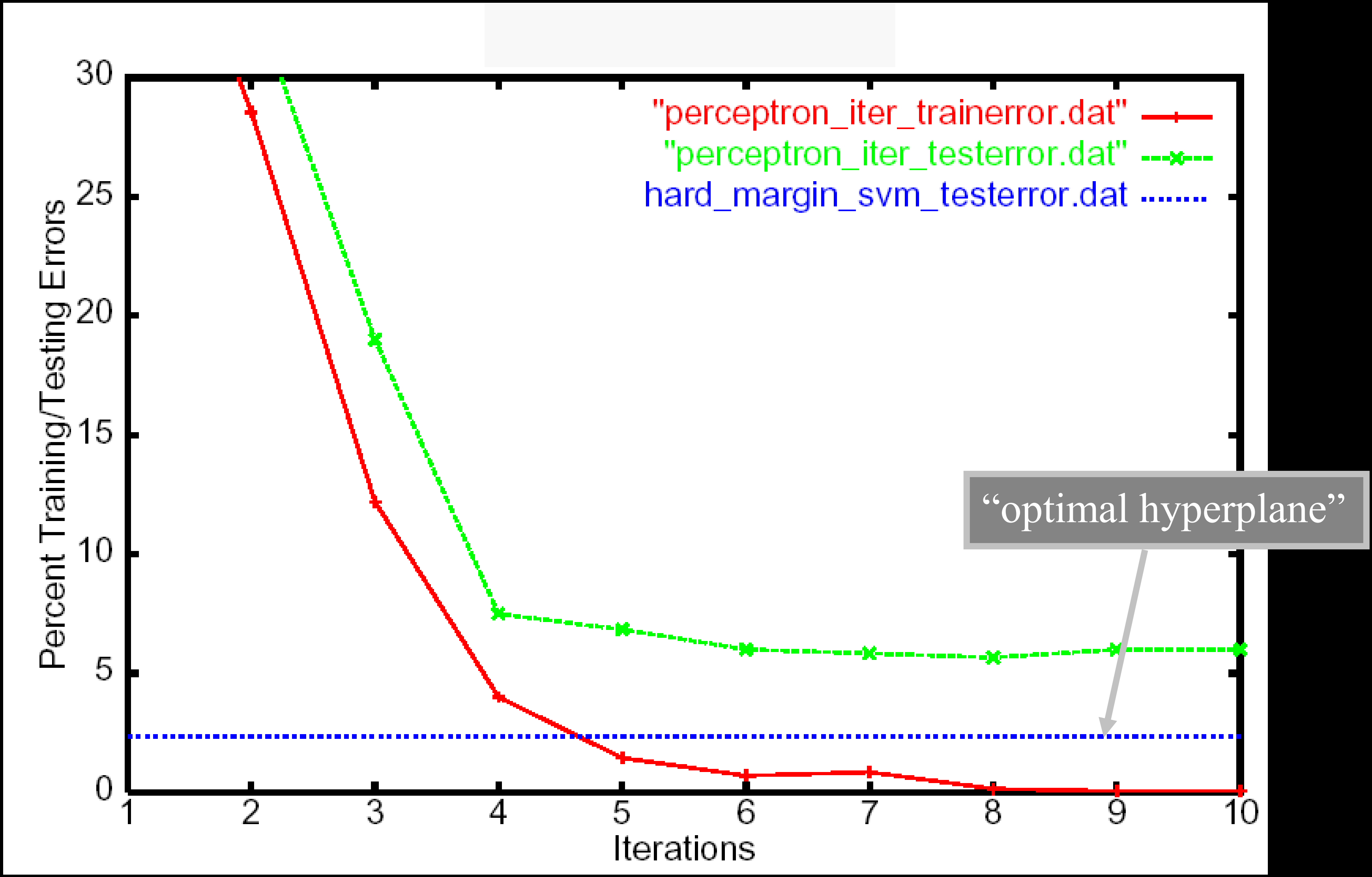
Algorithm:

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

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	$)$	$y_3 = -1$

Example: Reuters Text Classification



Online Learning Model

- Initialize hypothesis $h \in H$
- FOR i FROM 1 TO infinity
  - Receive x_i
  - Make prediction $\hat{y_i} = h(x_i)$
  - Receive true label y_i
  - Record if prediction was correct (e.g., $\hat{y_i} = y_i$)
  - Update h

→ Goal: minimize number of mistakes.

(Online) Perceptron Algorithm

- Input: $S = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$, $\vec{x}_i \in \Re^N$, $y_i \in \{-1, 1\}$
- Algorithm:
  - $\vec{w}_0 = \vec{0}$, $k = 0$
  - FOR i=1 TO n
    - IF $y_i(\vec{w}_k \cdot \vec{x}_i) \leq 0$ ### makes mistake
      - $\vec{w}_{k+1} = \vec{w}_k + y_i\vec{x}_i$
      - $k = k + 1$
    - ENDIF
  - ENDFOR
- Output: $\vec{w}_k$

Perceptron Mistake Bound

Theorem: For any sequence of training examples $S = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)$ with

$$R = \max \|\vec{x}_i\| \,,$$

if there exists a weight vector $\vec{w}_{opt}$ with $\|\vec{w}_{opt}\| = 1$ and

$$y_i \left(\vec{w}_{opt} \cdot \vec{x}_i\right) \geq \delta > 0$$

for all $1 \leq i \leq n$, then the Perceptron makes at most

$$\frac{R^2}{\delta^2}$$

mistakes.

Margin of a Linear Classifier

Definition: *For a linear classifier h_w, the* **margin** δ *of an example* $(\vec{x}, y)$ *with* $\vec{x} \in \Re^N$ *and* $y \in \{-1, +1\}$ *is* $\delta = y(\vec{w} \cdot \vec{x})$.

Definition: *The margin is called* geometric margin, *if* $||\vec{w}|| = 1$. *For general* $\vec{w}$, *the term* functional margin *is used to indicate that the norm of* $\vec{w}$ *is not necessarily* 1.

Definition: *The (hard) margin of an unbiased linear classifier* $h_{\vec{w}}$ *on a sample* S *is* $\delta = min_{(\vec{x},y) \in S} y(\vec{w} \cdot \vec{x})$.

Definition: *The (hard) margin of an unbiased linear classifier* $h_{\vec{w}}$ *on a task* $P(X, Y)$ *is*

$$\delta = inf_{S \sim P(X,Y)} min_{(\vec{x},y) \in S} y(\vec{w} \cdot \vec{x}).$$