

Online Algorithms

Outline

- Online Model: Mistake Bound
 - Simple algorithms
 - CON, HAL, ELIM
 - Online to PAC
- Linear Separator
 - Perceptron
 - Realizable case
 - Unrealizable case – Hinge loss
 - Winnow

Online Model

- Examples arrive sequentially
- Need to make a prediction
 - Afterwards observe the outcome
- No distributional assumptions
- **Goal: Minimize the number of mistakes**



Con Algorithm

- Realizable setting.
- Modest goal:
 - At most $|C|-1$ mistakes
- How?
- At time t
 - Let C_t the consistent concepts on $x_1, x_2 \dots x_{t-1}$
 $\forall c \in C_t \quad \forall i \leq t-1$
 $c(x_i) = c^*(x_i)$
 - Let $h_t \in C_t$
 - arbitrary
 - Predict $h_t(x_t)$
 - Observe $c^*(x_t)$

CON Algorithm

- **Theorem:** For any concept class C , CON makes the most $|C| - 1$ mistakes
- **Proof:** Initially $C_1 = C$.
- After each mistake $|C_t|$ decreases by at least 1
- $|C_t| \geq 1$, since $c^* \in C$ at any t
- Therefore number of mistakes is bound by
 $|C| - 1$



Can we do better ?!

HAL – halving algorithm

- Change selection of h_t
- At time t , given x_t
 - For every $c \in C_t$
 - Compute $c(x_t)$
 - Predict the majority
 - $h_t(x) = M\{c(x): c \in C_t\}$
 - M is the majority
 - Might be that $h_t \notin C_t$
- HAL algorithm
- At time t
 - Let C_t the consistent concepts on x_1, x_2, \dots, x_{t-1}
 - $\forall c \in C_t \forall i \leq t-1$
 $c(x_i) = c^*(x_i)$
 - Let $h_t = M\{c(x): c \in C_t\}$
 - Predict $h_t(x_t)$
 - Observe $c^*(x_t)$

HAL –halving algorithm

- **Theorem:** For any concept class C , HAL makes the most $\lfloor \log_2 |C| \rfloor$ mistakes
- **Proof:** Initially $C_1 = C$.
- After each mistake $|C_{t+1}| \leq \frac{1}{2} |C_t|$
 - majority of consistent concepts were wrong.
- Therefore number of mistakes is bound by $\lfloor \log_2 |C| \rfloor$



Example: Learning OR of literals

- Inputs: (x_1, \dots, x_n)
- Literals : z_i, \bar{z}_i
- OR functions:
 - $z_1 \vee \bar{z}_4 \vee z_7$
- Realizable case:
 - $c^*(z)$ is an OR
 - L^* are the literals
- ELIM algorithm:
 - Initialize:
 $L_1 = \{z_1, \bar{z}_1, \dots, z_n, \bar{z}_n\}$
 - Time t , receive
 $x_t = (x_1, \dots, x_n)$
 - Predict $OR(L_t, x_t)$
 - Receive $c^*(x_t)$
 - If $c^*(x_t)$ negative
 - delete from L_t any positive literals of z .

What is the MAXIMUM number of mistakes?

Analysis of ELIM

- Properties:
 - $L^* \subset L_1$
 - $L_t \subseteq L_{t-1}$
 - $L^* \subseteq L_t$
 - On mistake
 - $|L_t| \leq |L_{t-1}| - 1$
 - First mistake:
 - $|L_{t-1}| - |L_t| = n$
 - Later mistakes
 - $|L_{t-1}| - |L_t| \geq 1$
- No mistake when $c^*(x_t)=1$
 - $L^* \subseteq L_t$
- Number of mistakes
 - Initially $|L_1| = 2n$
 - First mistake:
 - Reduce by n
 - Each additional mistake
 - Reduce by at least 1
 - Maximum number of mistakes $n+1$

Regret Minimization

- What about non-realizable setting?

- We can guarantee:

- Sequence of length T ; average loss:

- $Loss(online) \leq \min_{c \in C} Loss(c) + \sqrt{\frac{\log |C|}{T}}$

- Best expert problem.

- $Regret = Loss(online) - \min_{c \in C} Loss(c)$

- Not in the course !

Mistake Bound and PAC

- Are they equivalent
 - Class C is PAC learnable iff
 - Class C has finite Mistake Bound
- **NO!**
 - Prefix on of $[0,1]$
 - PAC learnable
 - No finite mistake bound
 - Why?
- What about the other direction
 - Algorithm A learns C in mistake bound model
 - Does A learn C in PAC model?
- We will show A_{PAC} that PAC learns C
 - A_{PAC} uses A

Conservative Online Algorithm

- A is conservative if
 - for every sample x_t if $c^*(x_t) = h_t(x_t)$ then
 - $h_{t+1} = h_t$
- Conservative algorithm
 - Changes hypothesis only after mistakes
 - After mistakes always changes hypothesis.
- Goal: show how to convert any algorithm A to a conservative algorithm A'
 - Same mistake bound
- Basic idea:
 - Feed A only mistakes

Conservative Mistake Bound Algorithm

- Given A Define A'
 - Initially $h_0 = A(\emptyset)$
 - At time t:
 - Predict $h_t(x_t)$
 - IF $c^*(x_t) = h_t(x_t)$,
 - THEN $h_{t+1} = h_t$
 - ELSE
 - » $ER = ER || x_t$
 - » $h_{t+1} = A(ER)$
- For any input sequence
 - If A' makes M mistakes
 - Then A makes M mistakes on ER
- Claim: Mistake Bound of A and A' identical

Building A_{PAC}

- Given algorithm A
 - At most M mistakes
 - A is conservative
- A_{PAC} works in stages
 - At most M+1
- Stage i:
 - Run algorithm A
 - If makes error
 - Start stage i+1
 - No error for $m_i = \frac{1}{\epsilon} \log \frac{1}{\delta_i}$
 - Output h_i
- Termination
 - Each stage A makes an error
 - At most M completed stages
- Performance
 - If h_i is ϵ – bad prob output:
 - $(1 - \epsilon)^{m_i} \leq e^{-\epsilon m_i} = \delta_i$
- Sample size
 - $\sum_{i=1}^{M+1} m_i \leq \frac{M}{\epsilon} \left(\frac{M+1}{2} + \log \frac{1}{\delta} \right)$
- Confidence
 - Set: $\delta_i = \frac{\delta}{2^i}$
 - $\sum_i \delta_i \leq \delta$

Learning Linear Separators

- Input $\{0,1\}^d$ or \mathbb{R}^d
- Linear Separator
 - weights w in \mathbb{R}^d and threshold θ
 - hypothesis $h(x)=+1$ iff
$$\langle w, x \rangle = \sum w_i x_i \geq \theta$$
- Simplifying assumptions:
 - $\theta=0$ (add coordinate x_0 such that $x_0=1$ always)
 - $\|x\|=1$

Perceptron - Algorithm

- Initialize $\mathbf{w}_1 = (0, \dots, 0)$
- Given example \mathbf{x}_t ,
 - predict positive iff $\langle \mathbf{w}_t, \mathbf{x}_t \rangle \geq 0$
- On a Mistake t : $\mathbf{w}_{t+1} = \mathbf{w}_t + c_t(\mathbf{x}) \mathbf{x}_t$,
 - Mistake on negative (i.e., $c^*(\mathbf{x}) = +1$): $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}_t$.
 - Mistake on positive (i.e., $c^*(\mathbf{x}) = -1$): $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{x}_t$.

Perceptron - motivation

- **False Negative**

- $c_t(x) = +1$

- $\langle w_t, x_t \rangle$ negative

- after update

$$\langle w_{t+1}, x_t \rangle$$

$$= \langle w_t, x_t \rangle + \langle x_t, x_t \rangle$$

$$= \langle w_t, x_t \rangle + 1$$

- **False Positive**

- $c_t(x) = -1$

- $\langle w_t, x_t \rangle$ positive

- after update

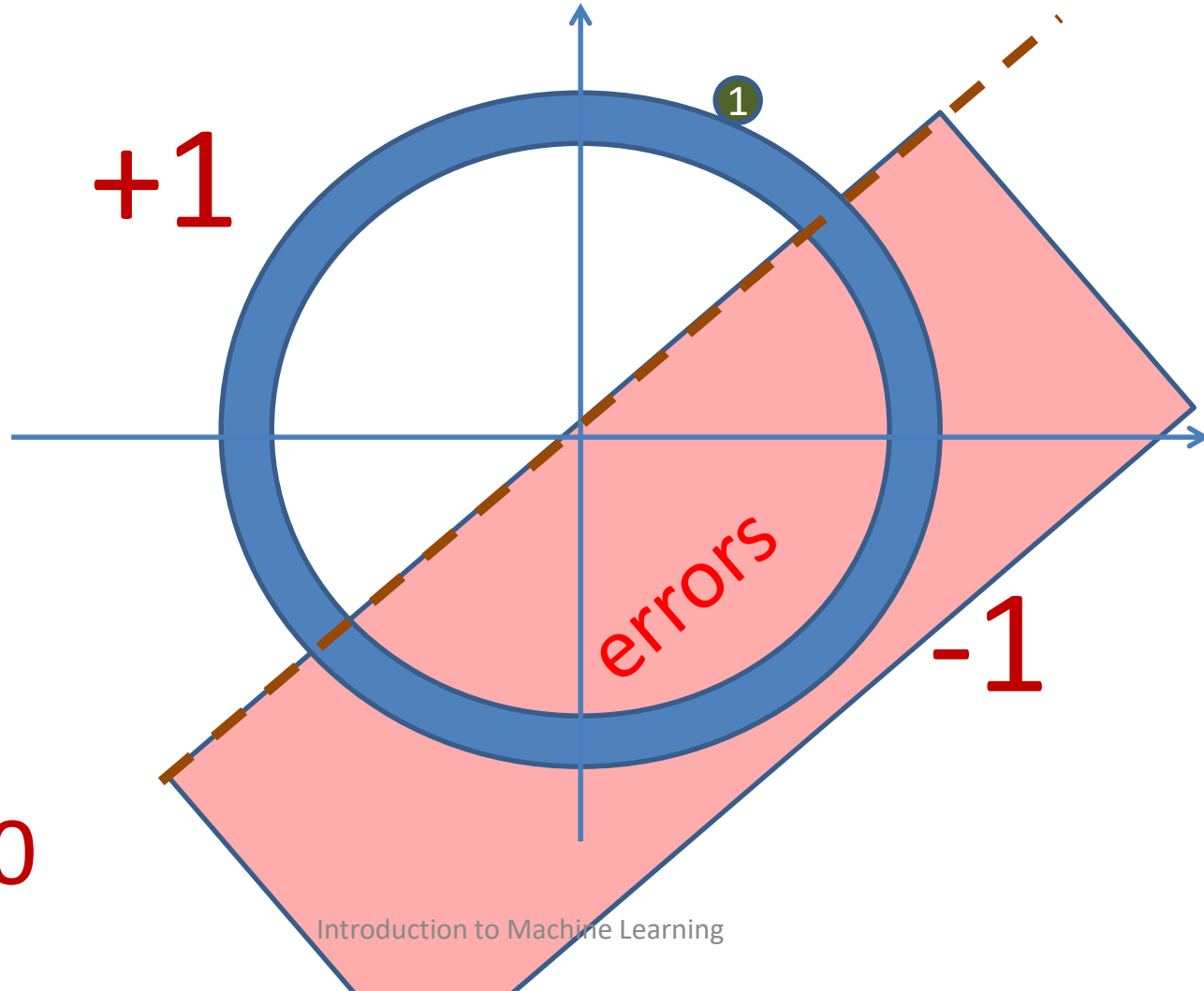
$$\langle w_{t+1}, x_t \rangle$$

$$= \langle w_t, x_t \rangle - \langle x_t, x_t \rangle$$

$$= \langle w_t, x_t \rangle - 1$$

Perceptron Example

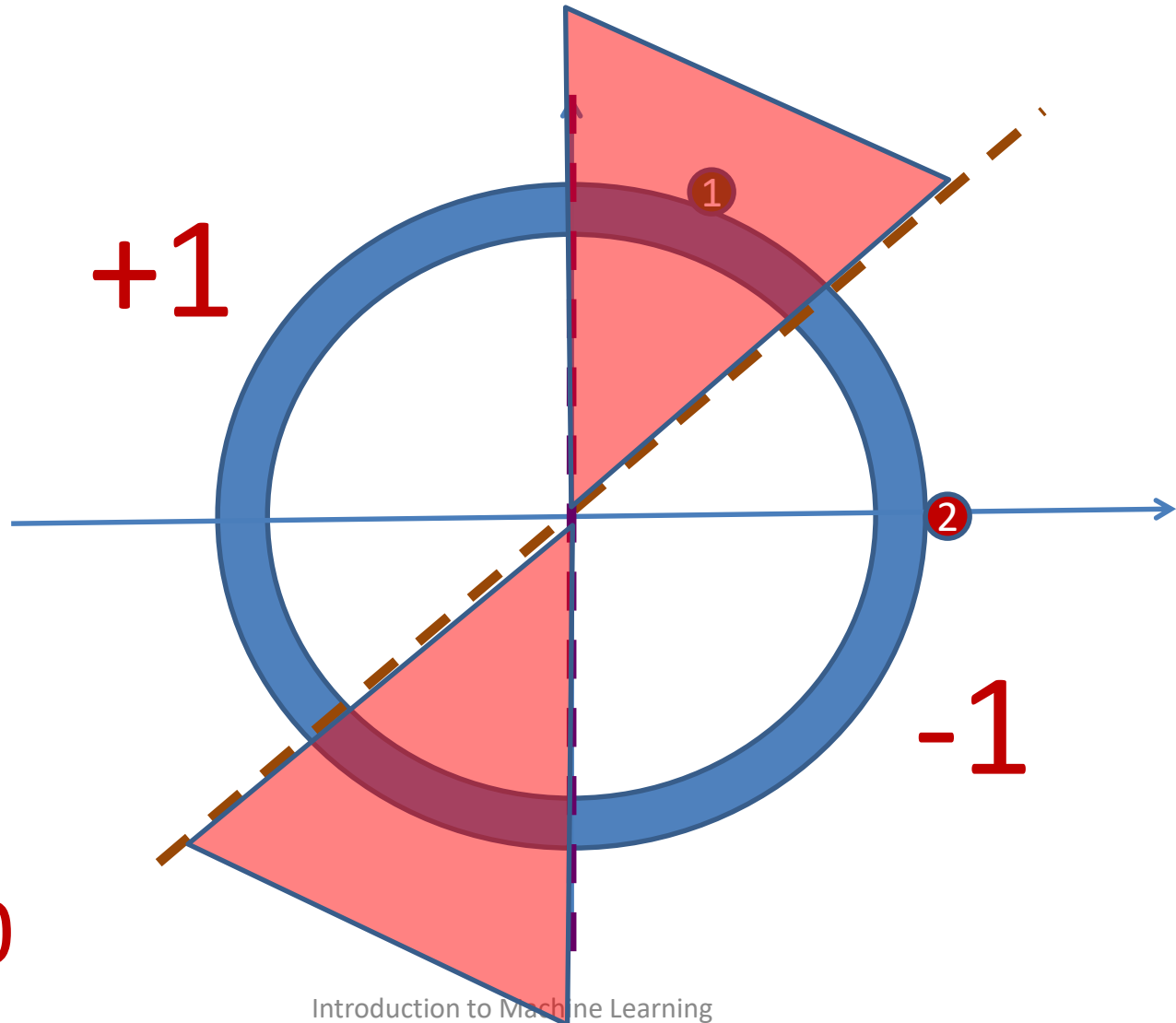
$$w_1 = (0,0)$$
$$w_2 = (0,0)$$



$$x_1 - x_2 = 0$$

Perceptron Example

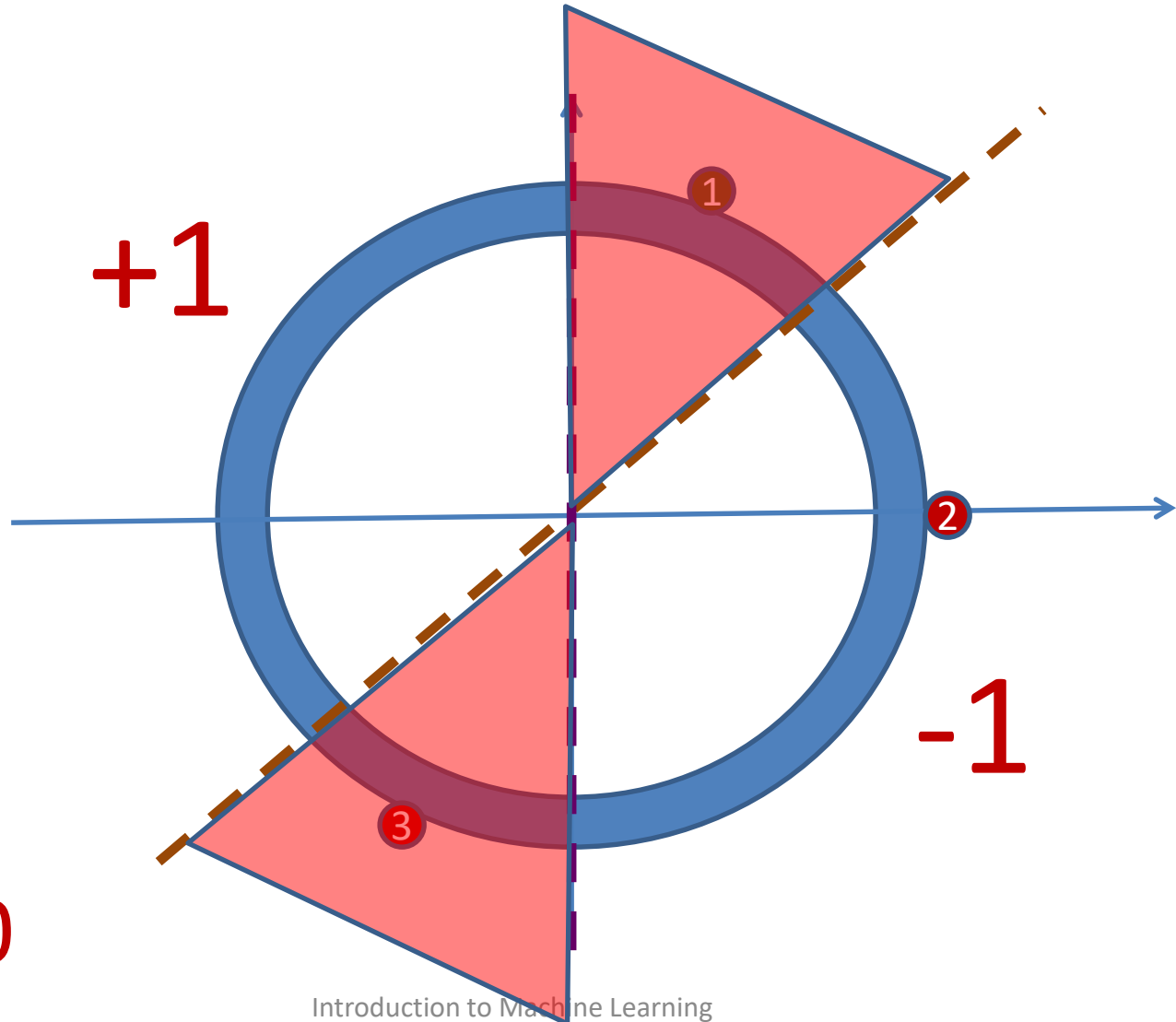
$$w_1 = (0,0)$$
$$w_2 = (0,0)$$
$$w_3 = (-1,0)$$



$$x_1 - x_2 = 0$$

Perceptron Example

$$\begin{aligned}w_1 &= (0,0) \\w_2 &= (0,0) \\w_3 &= (-1,0)\end{aligned}$$



$$x_1 - x_2 = 0$$

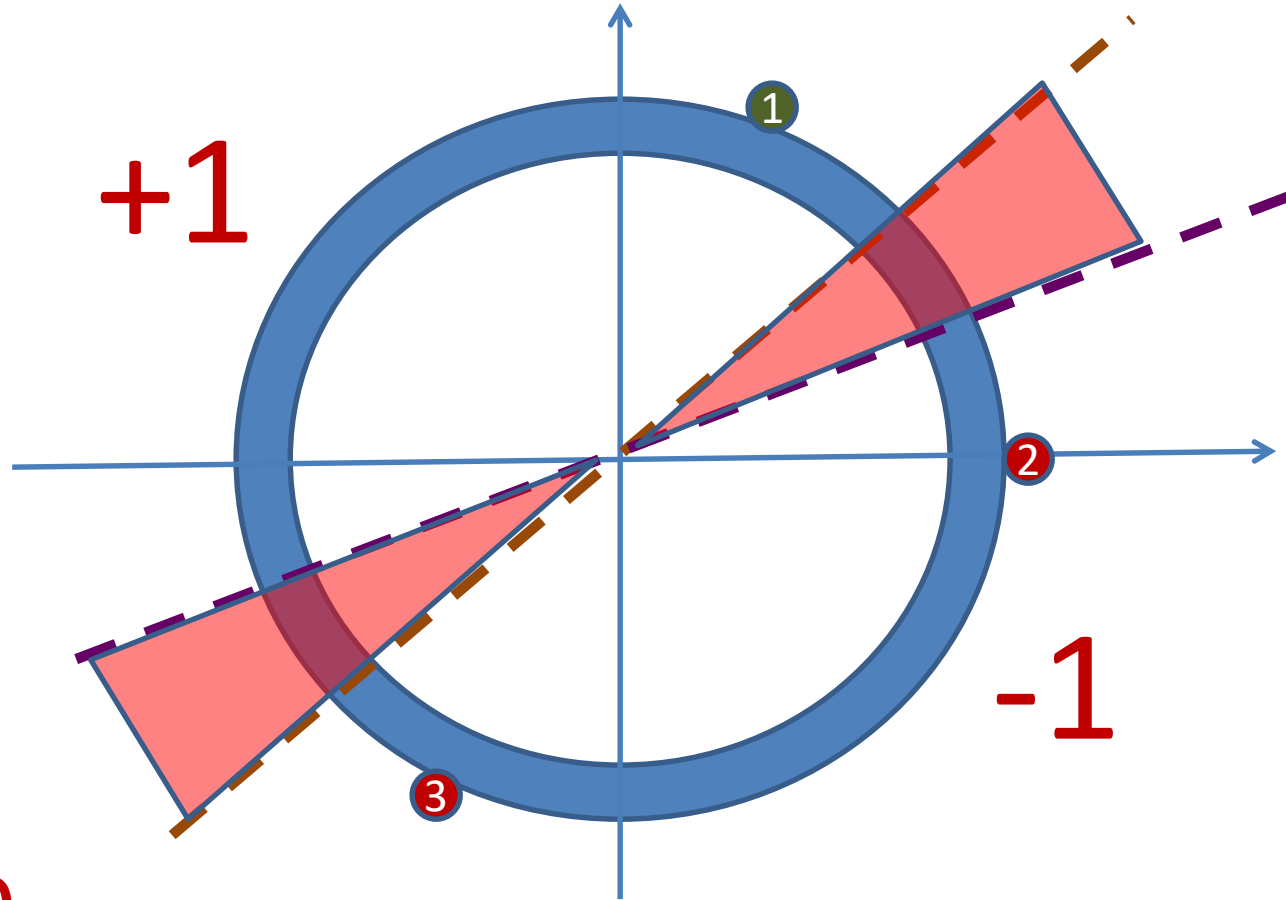
Perceptron Example

$$w_1 = (0,0)$$

$$w_2 = (0,0)$$

$$w_3 = (-1,0)$$

$$w_4 = (-0.2, +0.6)$$



$$x_1 - x_2 = 0$$

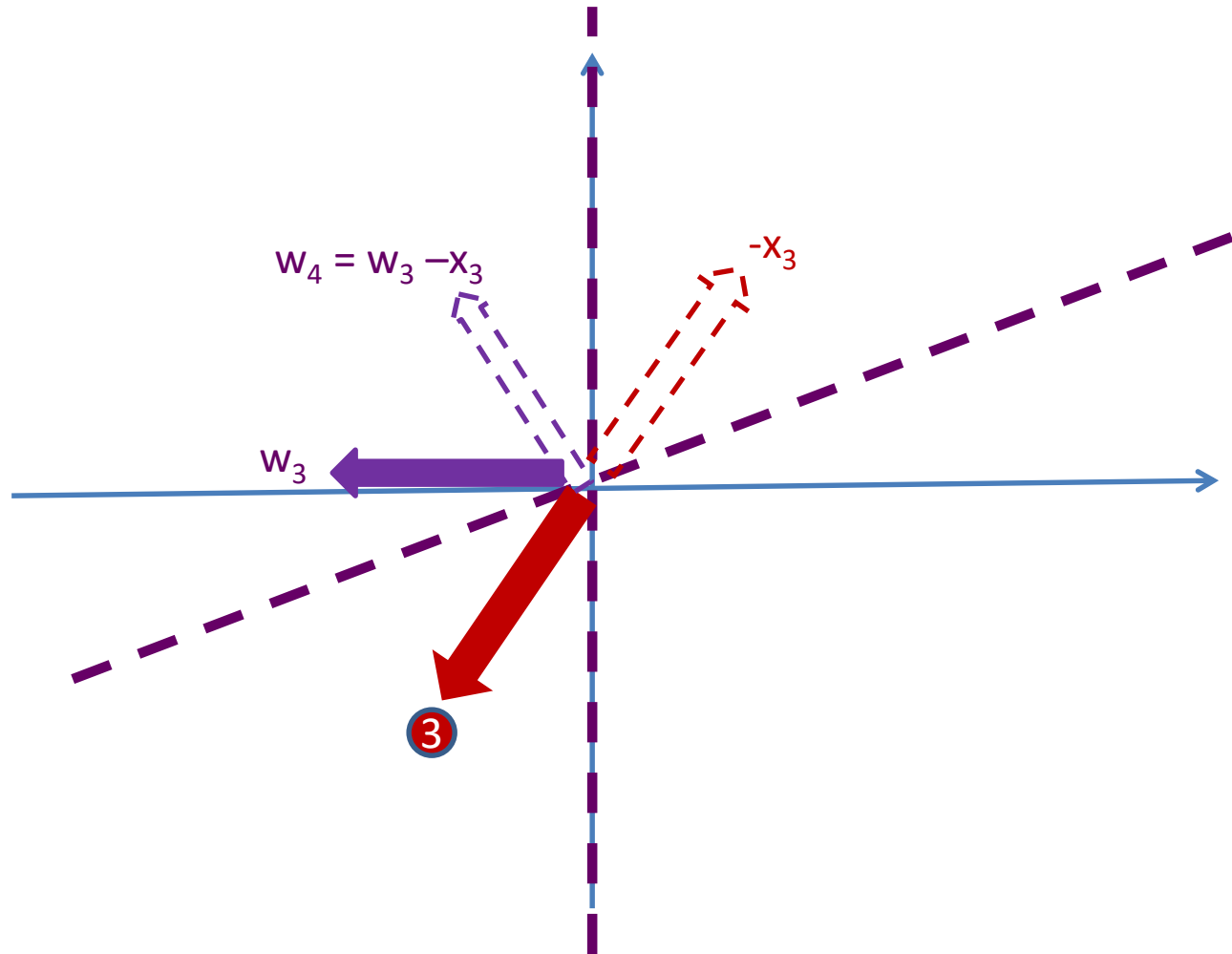
Perceptron - Geometric Interpretation

$$w_1 = (0,0)$$

$$w_2 = (0,0)$$

$$w_3 = (-1,0)$$

$$w_4 = (-0.2, +0.6)$$

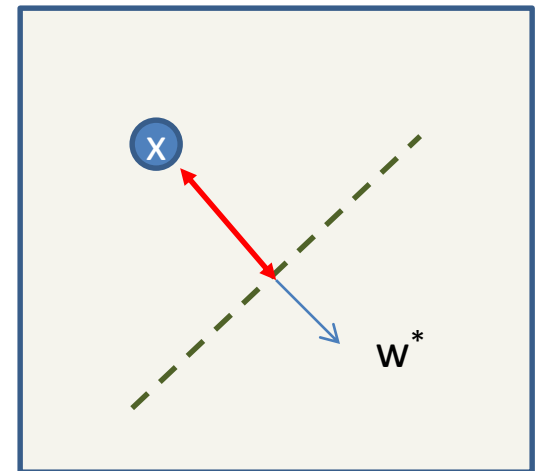


Perceptron - Analysis

- target concept $c^*(x)$ uses w^* and $\|w^*\| = 1$
- Margin γ :

➤ For any x in S

$$\gamma = \min_{x \in S} \frac{|\langle x, w^* \rangle|}{\|x\|}$$



- **Theorem:** *Number of mistakes $\leq 1/\gamma^2$*

Perceptron - Performance

Claim 1:

$$\langle w_{t+1}, w^* \rangle \geq \langle w_t, w^* \rangle + \gamma$$

Assume $c^*(x) = +1$

$$\langle w_{t+1}, w^* \rangle =$$

$$\langle (w_t + x), w^* \rangle =$$

$$\langle w_t, w^* \rangle + \langle x, w^* \rangle \geq$$

$$\langle w_t, w^* \rangle + \gamma$$

Similar for $c^*(x) = -1$

Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$

Assume $c^*(x) = +1$

$$\|w_{t+1}\|^2 =$$

$$\|w_t + x\|^2 =$$

$$\|w_t\|^2 + 2\langle w_t, x \rangle + \|x\|^2 \leq$$

$$\|w_t\|^2 + 1$$

Since x is a mistake $\langle w_t, x \rangle$ is negative.

Similar for $c^*(x) = -1$

Perceptron - performance

Claim 3: $\langle w_t, w^* \rangle \leq \|w_t\|$


$$\langle w_t, w^* \rangle \leq \langle w_t, \frac{w_t}{\|w_t\|} \rangle = \|w_t\|$$

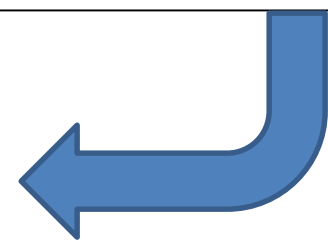
Completing the proof

• After M mistakes:

$$\langle w_{M+1}, w^* \rangle \geq \gamma M \quad (\text{claim 1})$$

$$\|w_{M+1}\|^2 \leq M \quad (\text{claim 2})$$


$$\gamma M \leq \langle w_{M+1}, w^* \rangle \leq \|w_{M+1}\| \leq \sqrt{M}$$


$$M \leq \frac{1}{\gamma^2}$$

Perceptron

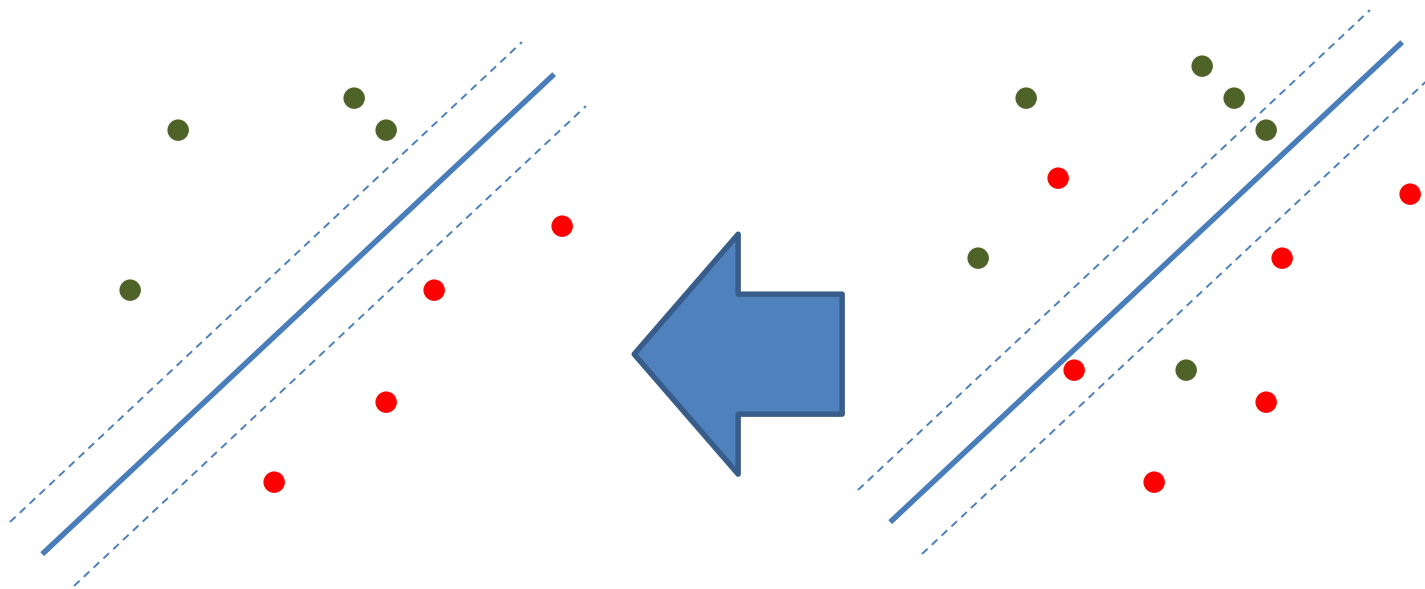
- Guaranteed convergence
 - realizable case
- Can be very slow (even for $\{0,1\}^d$)
- Additive increases:
 - problematic with large weights
- Still, a simple benchmark

Perceptron – Unrealizable case

Motivation

Realizable case

Unrealizable case



Hinge Loss

Motivation

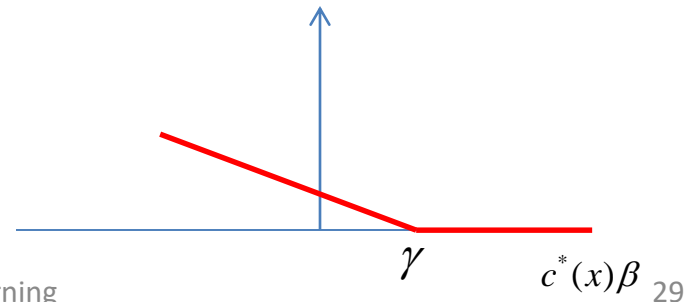
- “Move” points to be realizable
 - with margin γ
- correct points
 - both classification and margin
 - zero loss
- mistake points
 - even just margin
 - loss is the distance

Definition

- Assume $\langle x, w \rangle = \beta$
- Hinge Loss with margin γ :

$$\max\left\{0, 1 - \frac{c^*(x)\beta}{\gamma}\right\}$$

- Compare to Error: $c^*(x)\beta < 0$



Perceptron - Performance

- Let TD_γ = total distance
 $\sum_i \max\{0, \gamma - c^*(x)\beta_i\}$, where $\beta_i = \langle x_i, w^* \rangle$
- Claim 1': $\langle w_{M+1}, w^* \rangle \geq \gamma M - TD_\gamma$
- Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$
- Bounding the mistakes:

$$\sqrt{M} \geq \gamma M - TD_\gamma \quad \Rightarrow \quad M \leq \frac{1}{\gamma^2} + \frac{2}{\gamma} TD_\gamma$$

Winnow

Winnow –motivation

- Updates
 - multiplicative vs additive
- Domain
 - $\{0,1\}^d$ or $[0,1]^d$
 - we will use $\{0,1\}^d$
- Weights
 - non-negative
 - monotone function
- Separation
 - $c^*(x)=+1: \langle w^*, x \rangle \geq \theta$
 - $c^*(x)=-1: \langle w^*, x \rangle \leq \theta - \gamma$
 - $\theta \geq 1$
 - part of the input
- Remarks:
 - normalizing x in L_∞ to 1

Winnow - Algorithm

- parameter $\beta > 1$
 - we will use $\beta = 1 + \gamma/2$
- Initialize $\mathbf{w} = (\mathbf{1}, \dots, \mathbf{1})$
- predict $h(\mathbf{x}) = +1$ iff
$$\langle \mathbf{w}, \mathbf{x} \rangle \geq \theta$$
- For a mistake:
- False Positive (**demotion**)
 - $c^*(\mathbf{x}) = -1, h(\mathbf{x}) = +1$
 - for every $x_i = 1$: $w_i = w_i / \beta$
- False Negative (**promotion**)
 - $c^*(\mathbf{x}) = +1, h(\mathbf{x}) = -1$
 - for every $x_i = 1$: $w_i = \beta w_i$

Winnow - intuition

- Demotion step
 - target negative
 - hypothesis positive
- Before update
$$\langle w, x \rangle = \alpha \geq \theta$$
- After the update:
$$\langle w, x \rangle = \alpha / \beta < \alpha$$
- Decrease in $\sum w_i$
 - at least $(1 - \beta^{-1})\theta$
- Promotion step
 - target positive
 - hypothesis negative
- Before update
$$\langle w, x \rangle = \alpha < \theta$$
- After the update:
$$\langle w, x \rangle = \alpha \beta > \alpha$$
- Increase in $\sum w_i$
 - at most $(\beta - 1)\theta$

Winnow - example

- Target function:
- $w^*=(2,2,0,0)$
- $\theta=2$, $\beta=2$
- What is the target function?
 - $x_1 \vee x_2$
 - monotone OR
- $w_0=(1,1,1,1)$
- $x_1=(0,0,1,1)$ $c_t(x_1)=-1$
 - $w_1=(1,1, \frac{1}{2}, \frac{1}{2})$
- $x_2=(1,0,1,0)$ $c_t(x_2)=+1$
 - $w_2=(2,1, 1, \frac{1}{2})$
- $x_3=(0,1,0,1)$ $c_t(x_3)=+1$
 - $w_3=(2,2, 1, 1)$

Winnow - Theorem

- **Theorem** (realizable case)

Number of mistakes bounded by

$$O\left(\frac{1}{\gamma^2} \frac{d}{\theta} + \frac{\ln \theta}{\gamma^2} \sum_{i=1}^d w_i^*\right)$$

- **Corollary:** For $\theta=d$ we have $O\left(\frac{\ln d}{\gamma^2} \sum_{i=1}^d w_i^*\right)$

Winnow - Analysis

- Mistakes
 - u promotion steps
 - v demotion steps
 - mistakes = $u+v$

- Lemma 1:

due to sum equalities before, and since weights are positive

$$v \leq \frac{\beta}{\beta - 1} \frac{d}{\theta} + \beta u$$

- Lemma 2: $w_i \leq \beta\theta$

We won't promote unless w_i by itself is not enough to pass theta

- Lemma 3:
after u prom.
and v demo.
exists i

$$\log w_i \geq \frac{\theta u - (\theta - \gamma)v}{\sum_{i=1}^d w_i} \log \beta$$

- Proof of theorem

Winnow vs Perceptron

Perceptron

- Additive updates
 - slow for large d
 - slow large weights
- Non-monotone
 - natural
- Simple Algorithm
- Margin scale $L_2(w^*)L_2(x)$

Winnow

- Multiplicative updates
 - handles large d nicely
 - ok with large weights
- Monotone
 - need to make monotone
 - flip non-monotone attributes
- Simple Algorithm
- Margin scale $L_1(w^*)L_\infty(x)$
- Additional factor $\log d$
 - for $\theta=d$

Summary

Linear Separators

- Today: Perceptron and Winnow
- Next week: SVM
- 2 weeks: Kernels
- later: Adaboost

Brief history:

- Perceptron
 - Rosenblatt 1957
- Fell out of favor in 70s
 - representation issues
- Reemerged with Neural nets
 - late 80s early 90s
- Linear separators:
 - Adaboost and SVM
- The immediate future: deep learning