

CSC 580 Principles of Machine Learning

04 Linear Classification; Perceptron

Chicheng Zhang

Department of Computer Science



*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

Announcements

- Piazza review questions: students who answer them will get participation credit
- HW0 grading still in progress
- Office hour time: Weds 11am-12pm -> Mons 5pm-6pm starting next week – sorry about my error

Linear classifiers

- Example application: spam filtering using bag-of-words



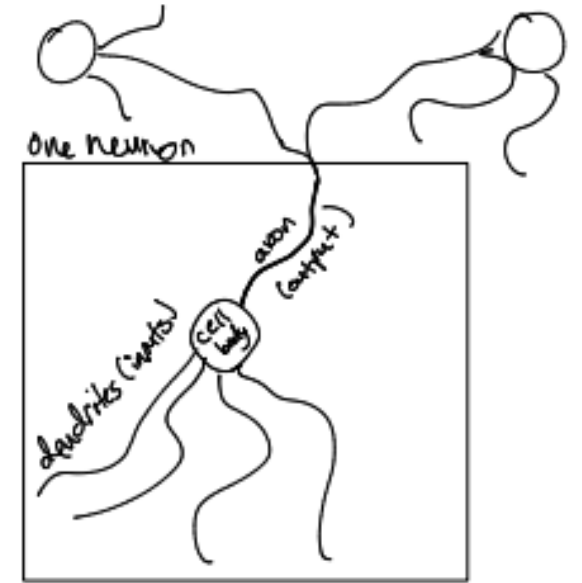
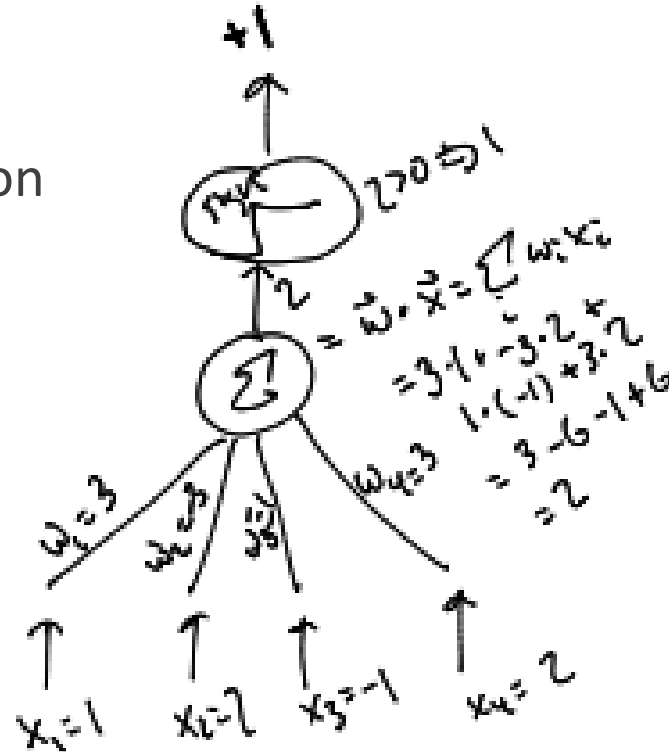
	free	offer	lecture	cs	Spam?
Email 1	1	1	0	0	+1
Email 2	0	0	1	1	-1

- If $0.124 \cdot x_{\text{free}} + 2.5 \cdot x_{\text{offer}} + \dots - 2.31 \cdot x_{\text{lecture}} > 2.12$ then
 - return “spam”
- else
 - return “nospam”
- end

Linear models: biological motivation

- Firing of a neuron depends on:
 - Whether the incoming neurons are firing
 - The strength of the connections
- The McCulloch-Pitts neural model:
a neuron implements a linear threshold function

$$h_w(x) = \text{sign}(\langle w, x \rangle)$$



Math review: inner product between vectors

- Given vector $u, v \in \mathbb{R}^d$,

$$\langle u, v \rangle = \sum_{i=1}^d u_i \cdot v_i$$

- Geometric interpretation:

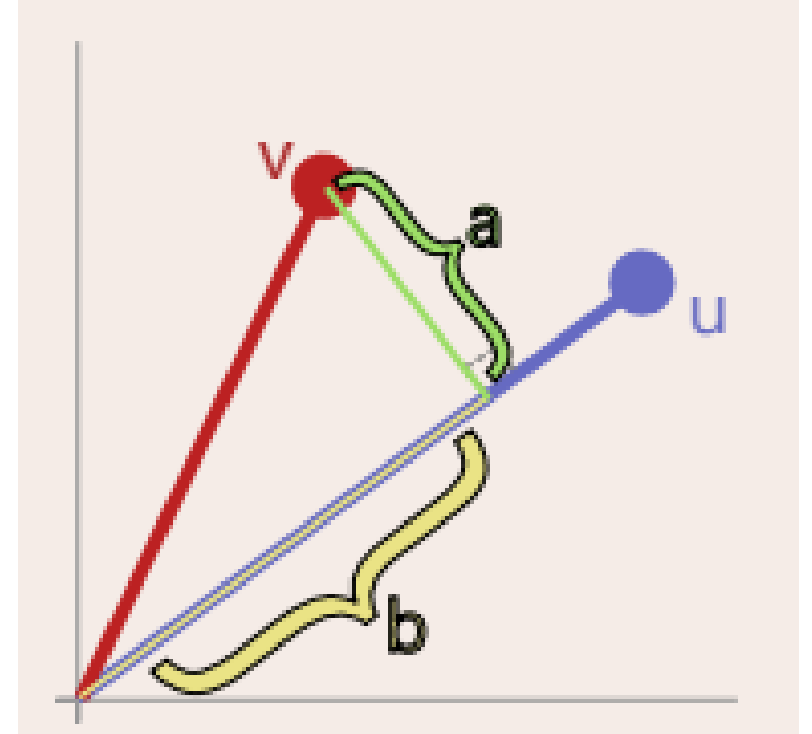
$$\langle u, v \rangle = \|u\|_2 \cdot \|v\|_2 \cdot \cos(\theta(u, v))$$

where $\theta(u, v) \in [0, \pi]$ is the angle between them

$\|v\|_2 \cdot \cos(\theta(u, v)) =$ (signed) length of v 's projection onto u

- Observe that $\cos(\theta(u, v)) \in [-1, +1]$

\Rightarrow Cauchy-Schwarz inequality: $\langle u, v \rangle \in [-\|u\|_2\|v\|_2, \|u\|_2\|v\|_2]$



Linear classifiers: geometric view

- Homogeneous linear classifier $h_w(x) = \text{sign}(\langle w, x \rangle)$
- Scale-insensitive
- Decision boundary: line in 2d, plane in 3d, hyperplane in general

- Non-homogeneous linear classifier $h_{w,b}(x) = \text{sign}(\langle w, x \rangle + b)$

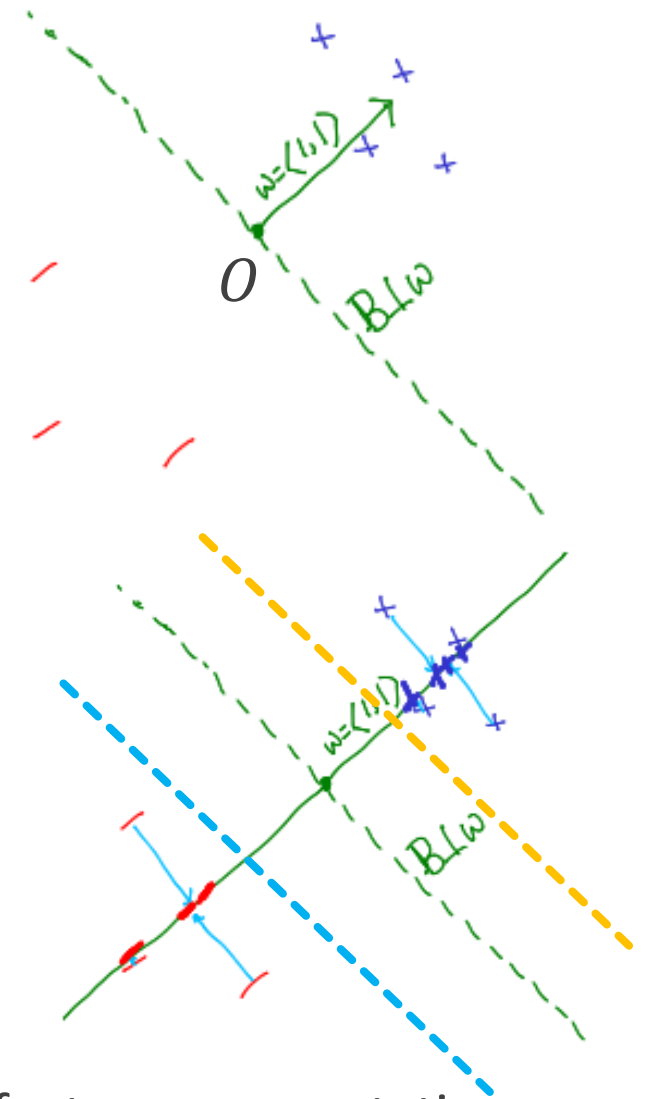
which decision boundary corresponds to offset $b > 0$? Blue or yellow?

- Sometimes convenient to view non-homogeneous. as homogeneous via feature augmentation

$$h_{w,b}(x) = \text{sign}(\langle (w, b), (x, 1) \rangle)$$

↓
 \tilde{w}

↓
 \tilde{x}



Training linear classifiers: The Perceptron algorithm (Rosenblatt, 1958)

- For training *homogeneous* linear classifiers



Initialize $w_1 \leftarrow (0, \dots, 0)$

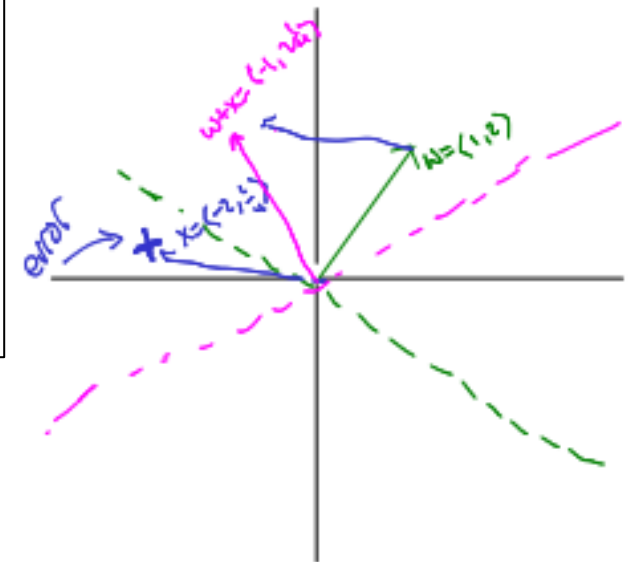
For $t = 1, 2, \dots, n$:

Process example $x_t \in \mathbb{R}^d$

Calculate prediction $\hat{y}_t = \text{sign}(w_t \cdot x_t)$

Update: if $\hat{y}_t = y_t$, $w_{t+1} \leftarrow w_t$;
otherwise, $w_{t+1} \leftarrow w_t + y_t x_t$.

	free	offer	lecture	cs	Spam?
Email 1	1	1	0	0	+1
Email 2	0	0	1	1	-1



- Properties: (1) Online (2) Error-driven

Perceptron for nonhomogeneous linear classifiers

- Idea: reduce to training homogeneous linear classifiers
- $h_{w,b}(x) = \text{sign}(\langle (w, b), (x, 1) \rangle) = \text{sign}(\langle \tilde{w}, \tilde{x} \rangle)$
- Multiple passes over the data

Algorithm 5 PERCEPTRONTRAIN(\mathbf{D} , $MaxIter$)

```
1:  $w_d \leftarrow 0$ , for all  $d = 1 \dots D$  // initialize weights
2:  $b \leftarrow 0$  // initialize bias
3: for  $iter = 1 \dots MaxIter$  do
4:   for all  $(x,y) \in \mathbf{D}$  do
5:      $a \leftarrow \sum_{d=1}^D w_d x_d + b$  // compute activation for this example
6:     if  $ya \leq 0$  then
7:        $w_d \leftarrow w_d + yx_d$ , for all  $d = 1 \dots D$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:     end if
10:  end for
11: end for
12: return  $w_0, w_1, \dots, w_D, b$ 
```

passes

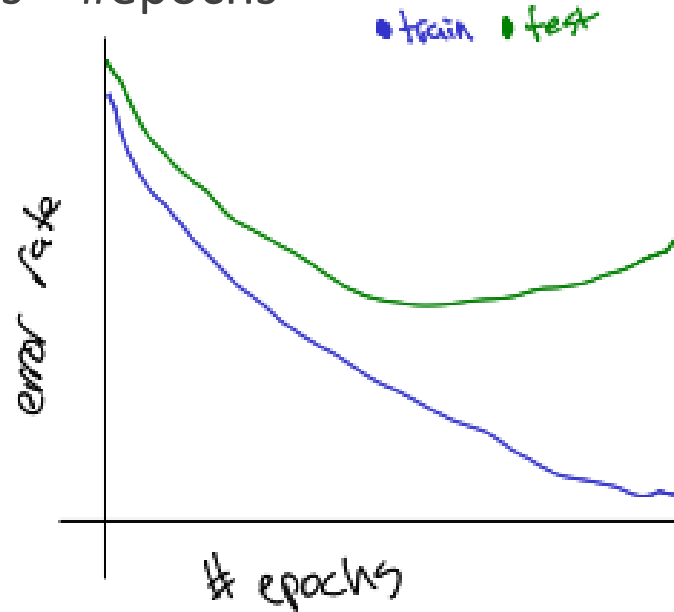
activation = decision
value

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

```
1:  $a \leftarrow \sum_{d=1}^D w_d \hat{x}_d + b$  // compute activation for the test example
2: return SIGN( $a$ )
```

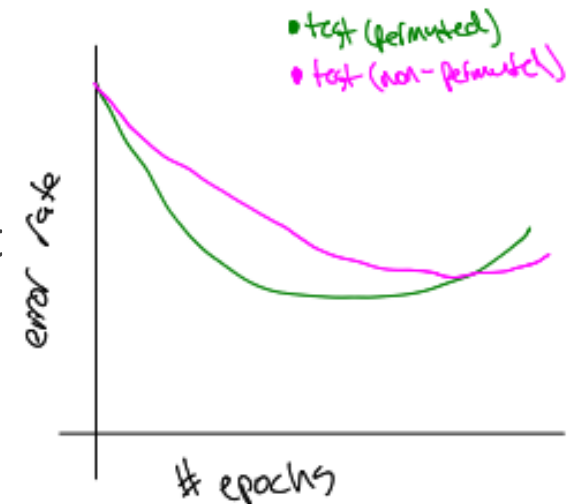
Perceptron: practical issues

- Hyperparameter: $\text{MaxIter} = \text{\#passes} = \text{\#epochs}$



- Data shuffling:

- A non-random training data sequence +++ ++ --- ---
- Drawback: only update using the first few examples in each segment
- Better: permute the data sequence for every pass



Perceptron: convergence properties

- Does the Perceptron's iterate w converge?
- Important notion: linear separability
- A dataset S is linearly separable if there exists w such that for all $(x, y) \in S$, $\text{sign}(\langle w, x \rangle) = y$

For iter = 1,2,....

For $(x, y) \in S$:

Calculate prediction $\hat{y} = \text{sign}(w \cdot x)$
if $\hat{y} \neq y$, $w \leftarrow w + y x$.

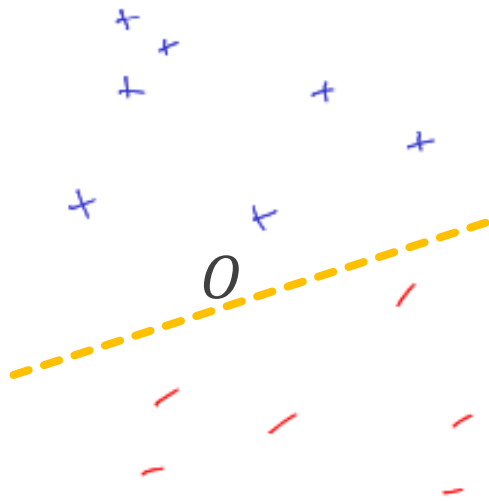


Figure 4.10: separable data

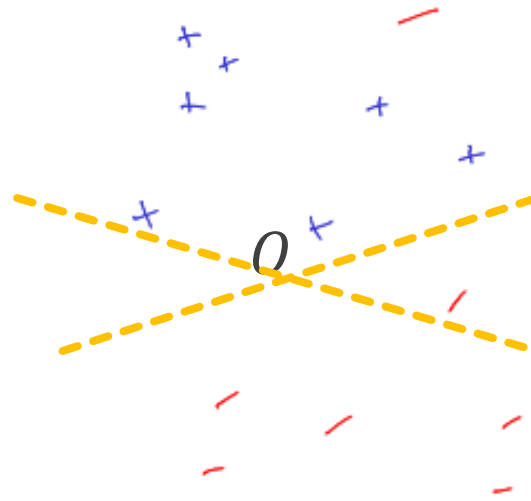


Figure 4.11: inseparable data

Observations:

- Inseparable c does not converge
- Separable \Rightarrow converge?

Q: how long does it take to converge?

Linear classification margins

- Measures easiness of a dataset for linear classification
- Easier dataset \Rightarrow faster convergence

- Margin of a linear classifier w on S :

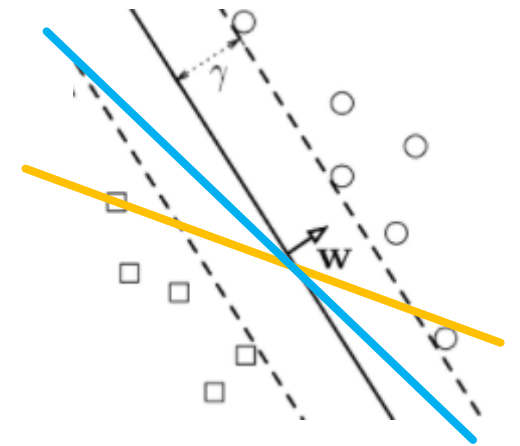
$$\text{margin}(S, w) = \begin{cases} \min_{(x,y) \in S} y \langle w, x \rangle, \\ -\infty, \end{cases}$$

w separates S
otherwise

- “Wiggle room” of w on S

- Margin of dataset S : $\text{margin}(S) = \max_{w: \|w\|_2=1} \text{margin}(S, w)$

- See book for definition of margins for nonhomogeneous linear classifiers

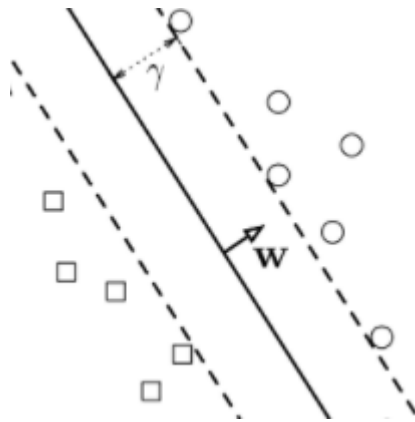


The Perceptron convergence theorem

Theorem (Perceptron Convergence Theorem, Novikoff 1962): Suppose the Perceptron algorithm is run on a dataset S ; Assume:

- $\text{margin}(S) \geq \gamma$, i.e. there exists w^* , $\|w^*\|_2 = 1$, $y\langle w^*, x \rangle \geq \gamma$ for all $(x, y) \in S$
- For all $(x, y) \in S$, $\|x\|_2 \leq 1$

then the Perceptron algorithm makes at most $1/\gamma^2$ updates throughout the process.



- Can also be phrased as an *online learning* mistake bound guarantee

Proof of Perceptron Convergence Theorem

- Denote $w^{(k)}$ the value of w after the k -th update; $w^{(0)} = (0, \dots, 0)$
- Idea: track the progression of $\langle w^{(k)}, w^* \rangle$ and $\|w^{(k)}\|_2$

- At the k -th update:

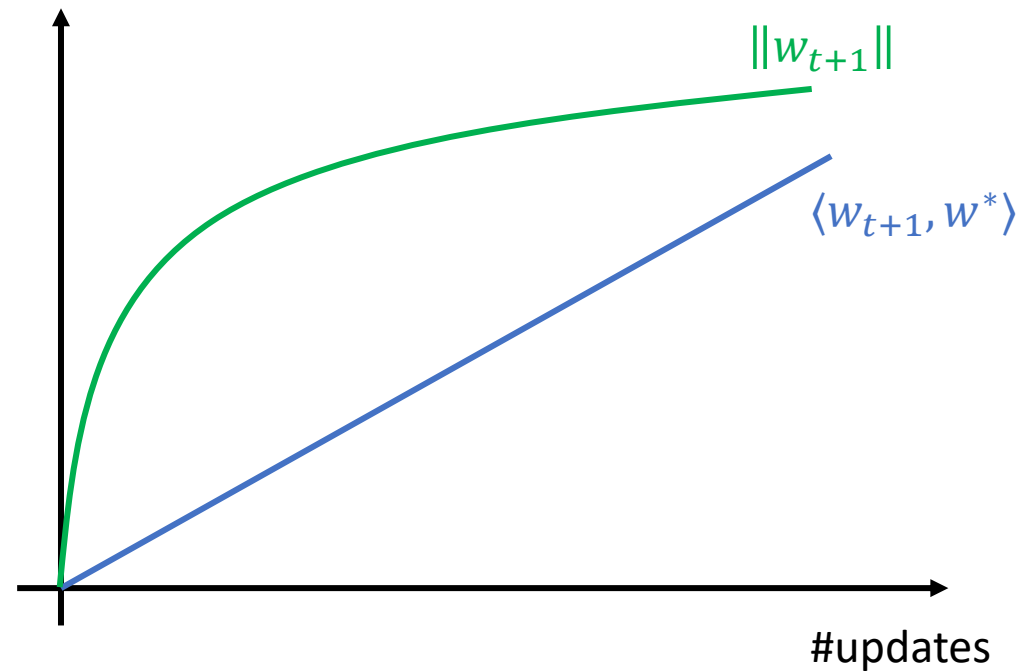
$$\langle w^{(k)}, w^* \rangle = \langle w^{(k-1)} + yx, w^* \rangle \geq \langle w^{(k-1)}, w^* \rangle + \gamma$$

$$\begin{aligned} \|w^{(k)}\|_2^2 &= \|w^{(k-1)} + yx\|_2^2 \\ &= \|w^{(k-1)}\|_2^2 + 2\langle w^{(k-1)}, yx \rangle + \|x\|_2^2 \\ &\leq \|w^{(k-1)}\|_2^2 + 1 \end{aligned}$$

Proof of Perceptron Convergence Theorem

- Therefore, if a total of k mistakes are made, then:

$$\langle w^{(k)}, w^* \rangle \geq k \gamma, \text{ and } \|w^{(k)}\| \leq \sqrt{k}$$



Proof of Perceptron Convergence Theorem

- Let $M = \#$ mistakes made up to time step n

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

- Meanwhile, by Cauchy-Schwarz,

$$\langle w_{n+1}, w^* \rangle \leq \|w_{n+1}\| \cdot \|w^*\| = \|w_{n+1}\|$$

- This implies that $M \gamma \leq \sqrt{M} \Rightarrow M \leq 1/\gamma^2$
- This holds for all n , which concludes the proof

Practical versions: voting Perceptron

- Naïve Perceptron: return the last iterate $w^{(K)}$
- Drawback:
 - say making one pass, last example is an outlier
 - Last update may ruin a previously trained good model

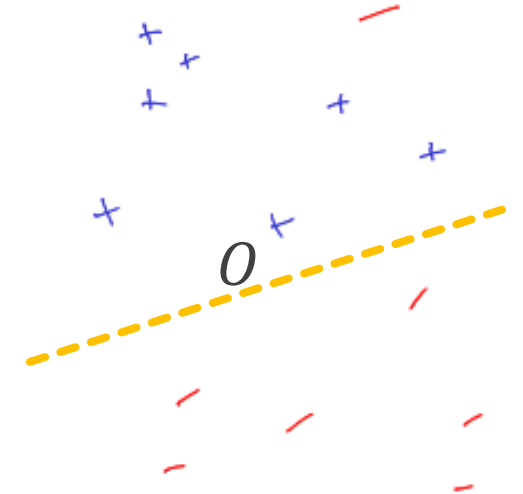


Figure 4.11: inseparable data

- A more robust output classifier:

$$h(x) = \text{sign} \left(\sum_{t=1}^T h_t(x) \right) = \text{sign} \left(\sum_{k=0}^K c^{(k)} h_{w^{(k)}}(x) \right)$$

\downarrow Linear classifier at iteration t \downarrow Number of times t when $h_t = h_{w^{(k)}}$

\uparrow $\in \{-1, +1\}$

- Has good predictive performance, but computationally expensive to maintain

Practical versions: averaged Perceptron

- $h(x) = \text{sign}(\langle \bar{w}, x \rangle)$, where $\bar{w} = \frac{1}{\sum_{k=0}^K c^{(k)}} \sum_{k=0}^K c^{(k)} w^{(k)}$ is the averaged predictor

- This is equivalent to $\text{sign}(\langle \sum_{k=0}^K c^{(k)} w^{(k)}, x \rangle)$

- Efficient implementation
(avoid extensive bookkeeping when no update)

- Exercise: show that the final output is \bar{w}

Algorithm 7 AVERAGEDPERCEPTRONTRAIN(\mathbf{D} , $MaxIter$)

```

1:  $w \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $b \leftarrow 0$  // initialize weights and bias
2:  $u \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $\beta \leftarrow 0$  // initialize cached weights and bias
3:  $c \leftarrow 1$  // initialize example counter to one
4: for  $iter = 1 \dots MaxIter$  do
5:   for all  $(x, y) \in \mathbf{D}$  do
6:     if  $y(w \cdot x + b) \leq 0$  then
7:        $w \leftarrow w + yx$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:        $u \leftarrow u + yc x$  // update cached weights
10:       $\beta \leftarrow \beta + yc$  // update cached bias
11:     end if
12:      $c \leftarrow c + 1$  // increment counter regardless of update
13:   end for
14: end for
15: return  $w - \frac{1}{c} u, b - \frac{1}{c} \beta$  // return averaged weights and bias

```

$$\sum_{k=0}^K z^{(k)} \sum_{k=0}^K c^{(k)} \quad \sum_{k=0}^K z^{(k)} \sum_{l < k} c^{(l)}$$

Perceptron: limitations

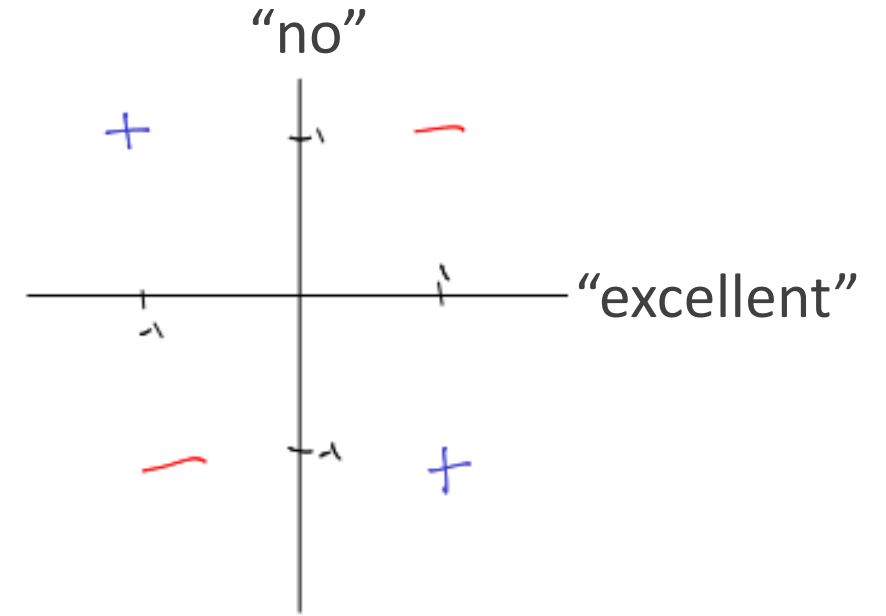
- The 'XOR' problem: data linearly nonseparable

- E.g. sentiment analysis

- Possible fix: introduce nonlinear feature maps

$$x = (x_1, x_2) \mapsto \phi(x) = (x_1, x_2, x_1x_2, x_1^2, x_2^2), \text{ e.g. containing "mega-feature" } x_{\text{no}} \cdot x_{\text{excellent}}$$

- Later in the course: kernel methods (high/infinite dim ϕ); neural networks (automatically learn ϕ)



Next lecture (9/12)

- Practical issues: feature selection; feature transformation; model performance evaluation
- Assigned reading: CIML Sections 5.1-5.6