CSC 580 Principles of Machine Learning

06 Linear Classification; Perceptron

Jason Pacheco

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 lecture slides by Chicheng Zhang & Kwang-Sung Jun

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

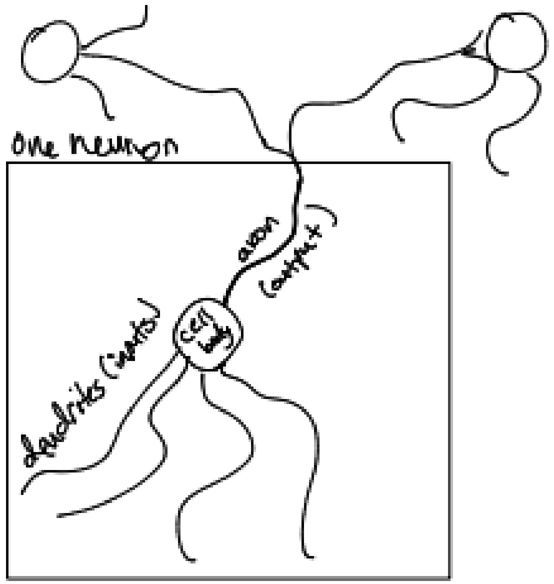
Linear models: biological motivation

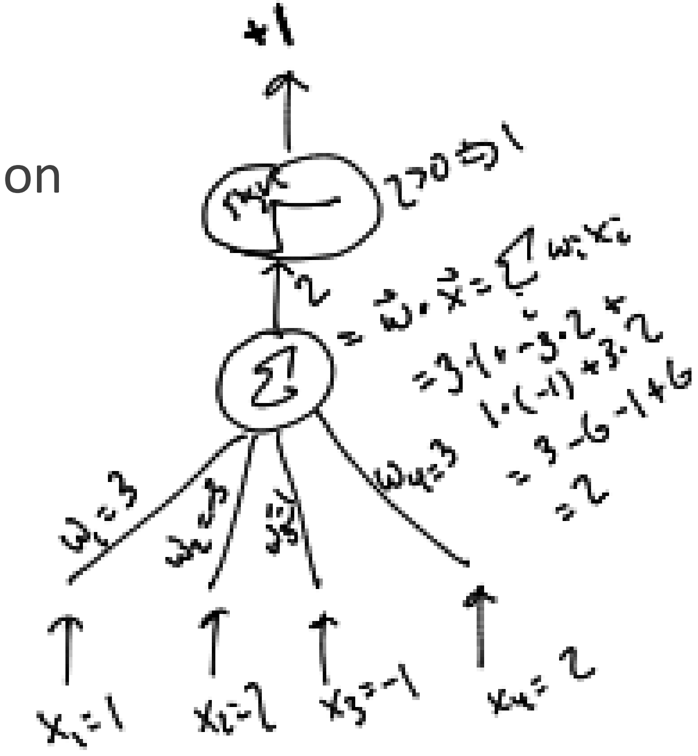
- Firing of a neuron depends on:
  - Whether the incoming neurons are firing
  - The strength of the connections

- 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 $u, v \in \mathrm{R}^d$,

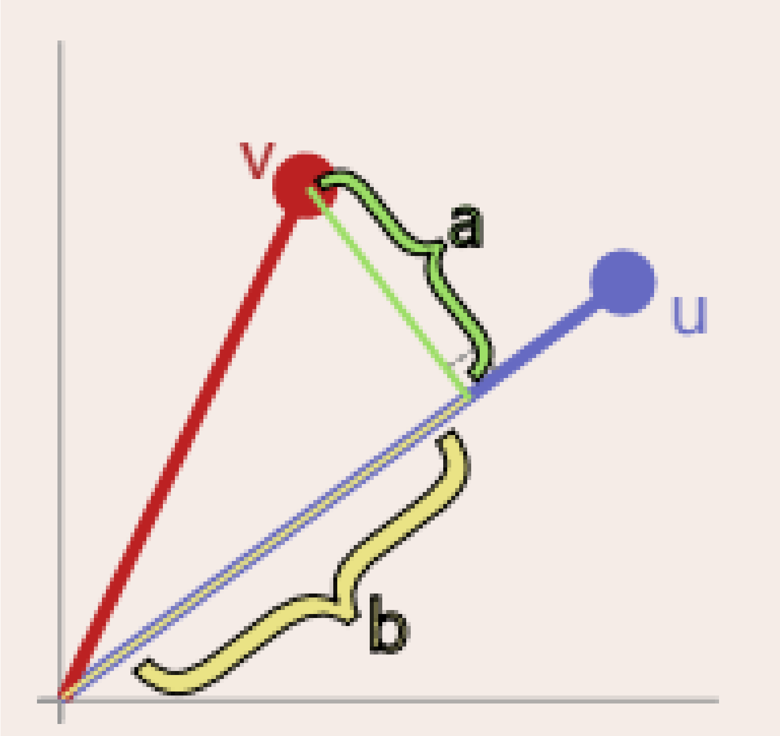
$$\langle u, v \rangle = \sum_{i=1}^{d} u_i \cdot v_i$$

- 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

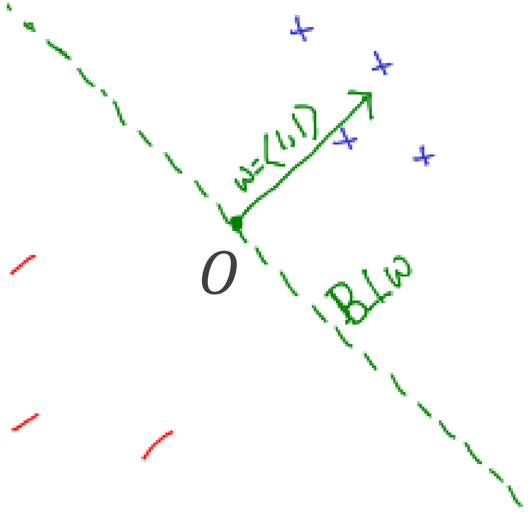
$||v||_2 \cdot \cos(\theta(u, v))$ = (signed) length of v's projection onto u



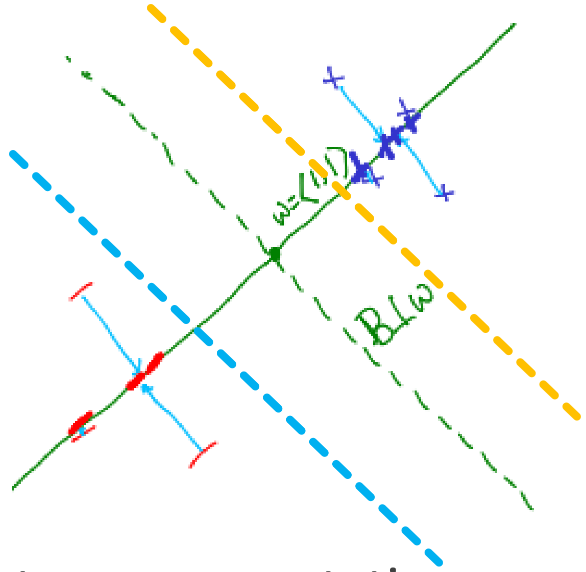
- Observe that $\cos\big(\theta(u, v)\big) \in [-1, +1]$

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

Linear classifiers: geometric view



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



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

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

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

$$h_{w,b}(x) = \mathrm{sign}(\langle \underbrace{(w,b)}_{\tilde{w}}, \underbrace{(x,1)}_{\tilde{x}} \rangle)$$

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

- For training *homogeneous* linear classifiers

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

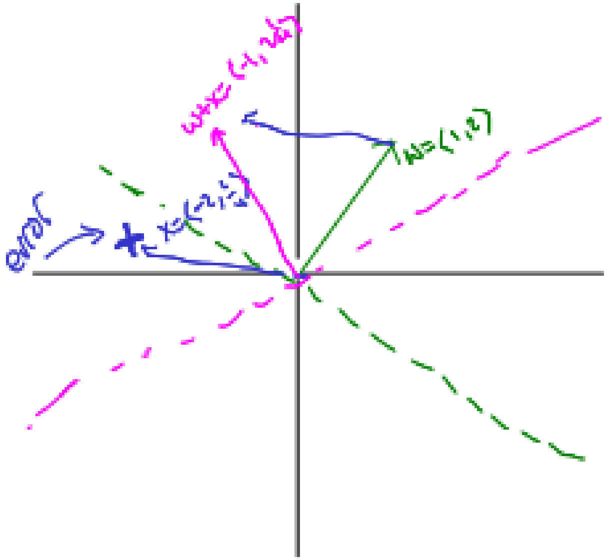
For $t = 1,2, \ldots, n$:

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

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

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*)

```
w_d ← 0, for all d = 1 ... D                         // initialize weights
b ← 0                                                 // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b                   // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D    // update weights
            b ← b + y                                 // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

passes

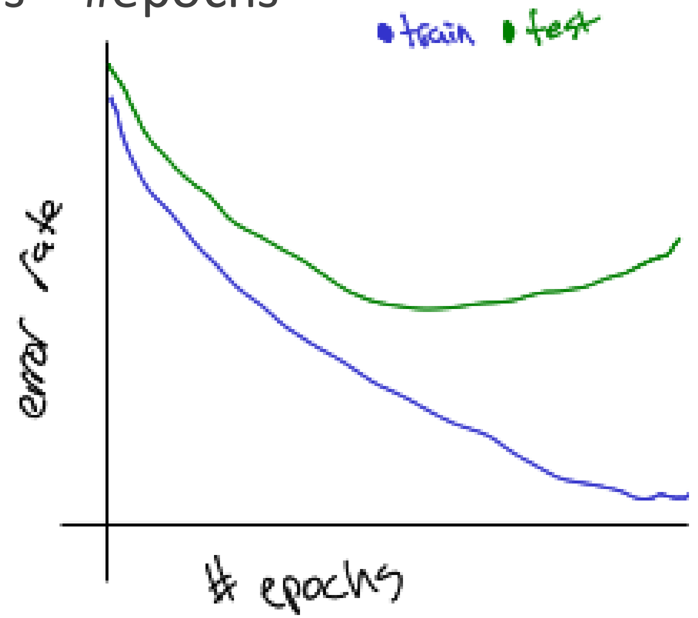
activation = decision value

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

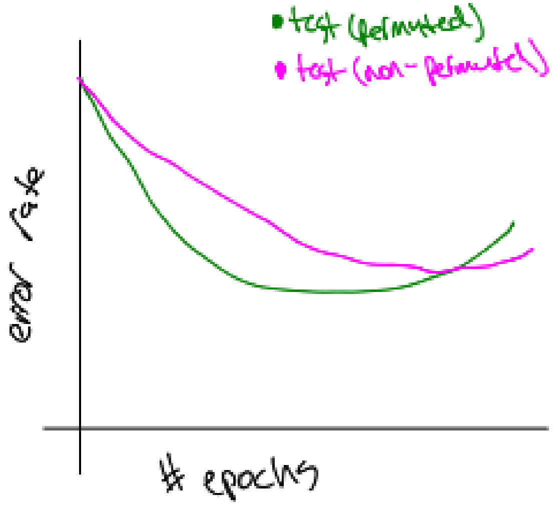
```
a ← Σ_{d=1}^{D} w_d x̂_d + b                          // compute activation for the test example
return SIGN(a)
```

Perceptron: practical issues

- Hyperparameter: MaxIter = #passes = #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

Question:
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 $(x, y) \in S$, $\mathrm{sign}(\langle w, x \rangle) = y$

For iter = 1,2,….
  For $(x, y) \in S$:
    Calculate prediction $\hat{y} = \mathrm{sign}(w \cdot x)$
    if $\hat{y} \neq y$, $w \leftarrow w + y\, x$.

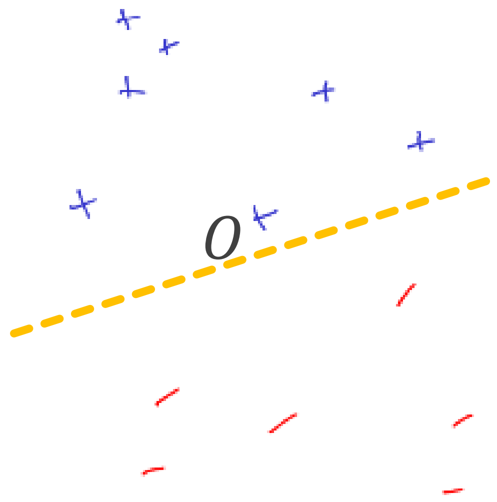


Figure 4.10: separable data

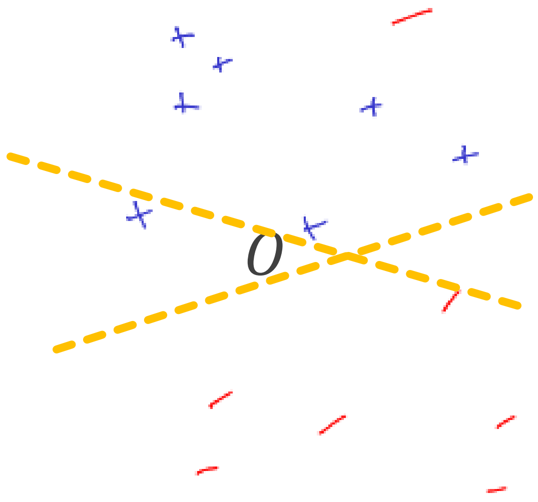


Figure 4.11: inseparable data

Observations:
- Inseparable c does not converge
- Separable $\Rightarrow$ converge?

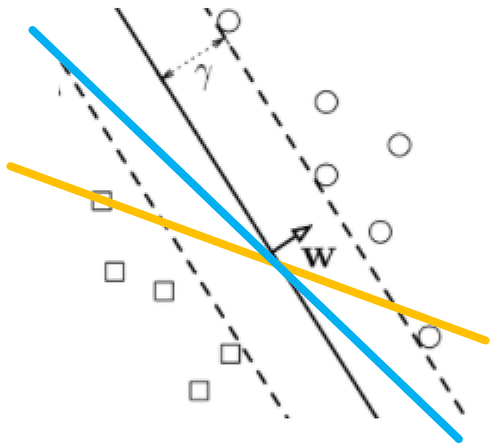
Q: how long does it take to converge?

Linear classification margins

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

- Margin of a linear classifier w on S:

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



- "Wiggle room" of w on S

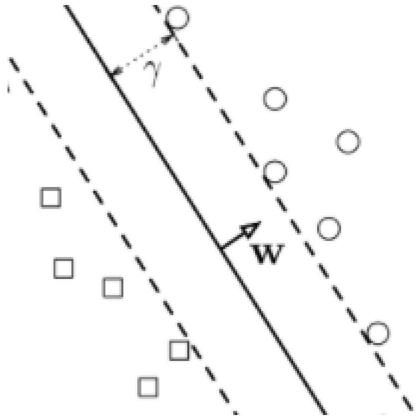
- Margin of dataset S: $\mathrm{margin}(S) = \max_{w : ||w||_2 = 1} \mathrm{margin}(S, w)$
- See book for definition of margins for nonhomogeneous linear classifiers

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 $\left\| w^{(k)} \right\|_2$

- At the k-th update:

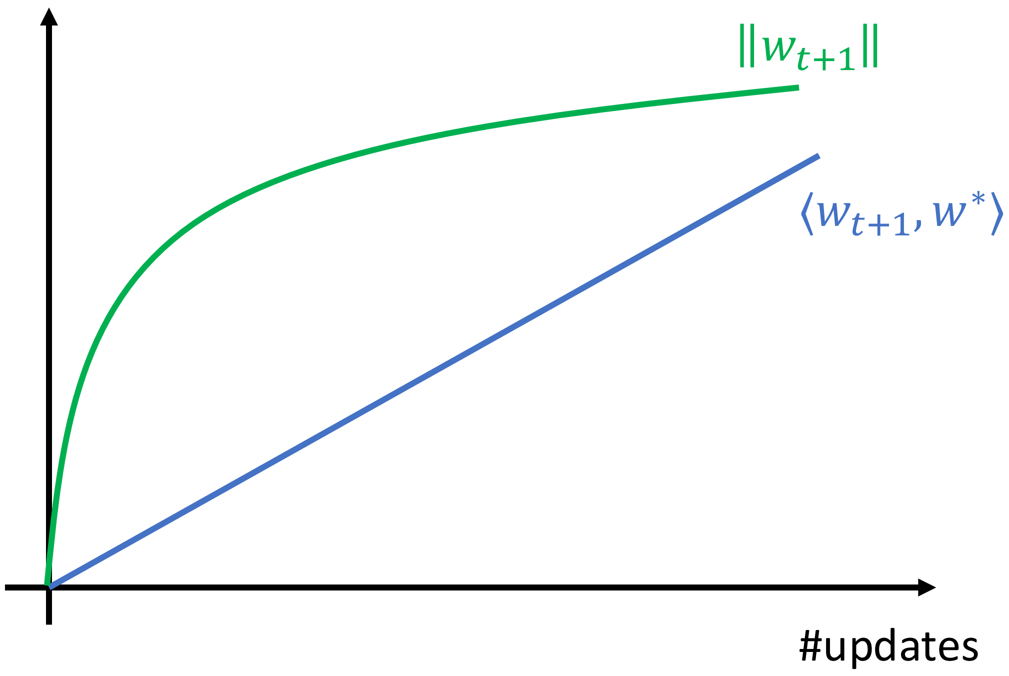
$$\langle w^{(k)}, w^* \rangle = \langle w^{(k-1)} + yx, w^* \rangle \geq \langle w^{(k-1)}, w^* \rangle + \gamma$$

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

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

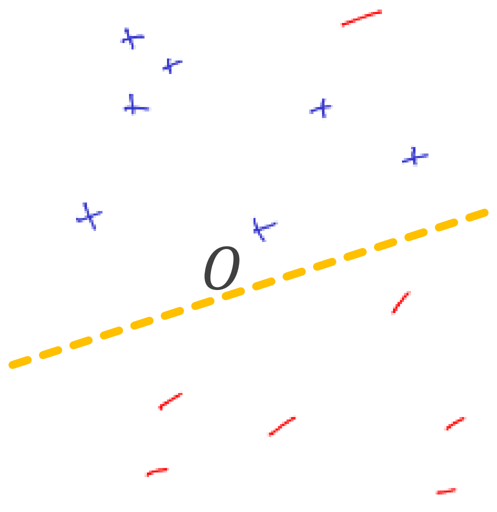


Figure 4.11: inseparable data

- A more robust output classifier:

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

$h_{w^{(k)}}(x) \in \{-1, +1\}$

$h_t(x)$: Linear classifier at iteration t

$c^{(k)}$: Number of times t when $h_t = h_{w^{(k)}}$

- Has good predictive performance, but computationally expensive to maintain

Practical versions: averaged Perceptron

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

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

- Efficient implementation

  (avoid extensive bookkeeping when no update)

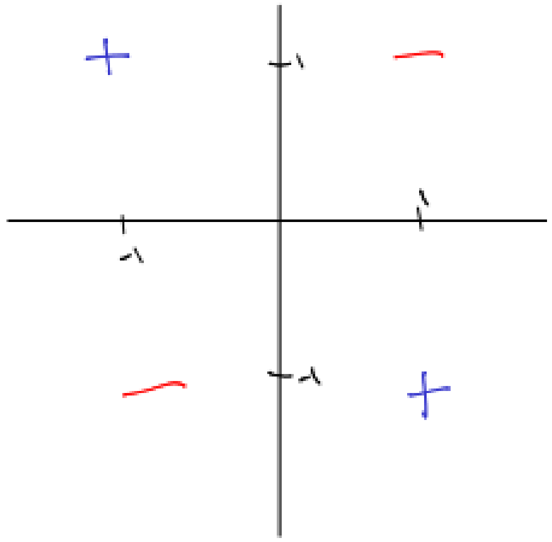
- Exercise: show that the final output is $\overline{w}$

Algorithm 7 AveragedPerceptronTrain(**D**, *MaxIter*)

```
w ← ⟨0, 0, . . . 0⟩  ,  b ← 0                    // initialize weights and bias
u ← ⟨0, 0, . . . 0⟩  ,  β ← 0                    // initialize cached weights and bias
c ← 1                                            // initialize example counter to one
for iter = 1 . . . MaxIter do
   for all (x,y) ∈ D do
      if y(w · x + b) ≤ 0 then
         w ← w + [y x]   → z^(k)                 // update weights
         b ← b + y                               // update bias
         u ← u + y c x                           // update cached weights
         β ← β + y c                             // update cached bias
      end if
      c ← c + 1                                  // increment counter regardless of update
   end for
end for
return w - (1/c) u, b - (1/c) β                  // return averaged weights and bias
```

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

Perceptron: limitations



- The 'XOR' problem: data linearly nonseparable
- E.g. sentiment analysis
- Possible fix: introduce nonlinear feature maps

  $x = (x_1, x_2) \mapsto \phi(x) = (x_1,\ x_2,\ x_1 x_2,\ x_1^2,\ x_2^2)$, 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)