# Week 2: Learning to Answer Yes/No

## Table of Contents



### Perceptron Hypothesis Set

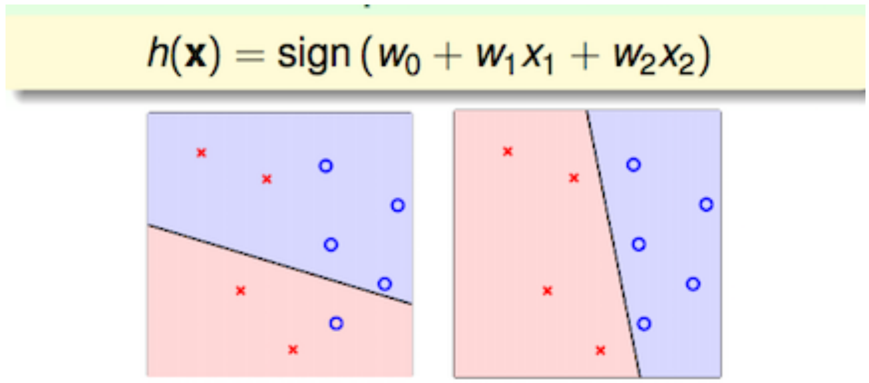
1. Perceptron Hypothesis
    - Two prediction outcomes from the hypothesis: yes(+), and no (-)
    - Vector form of perceptron hypothesis
        - $x_i \in D$: Input vector
        - $w_i$: Weight vector (magnitude determines importance of the variable, sign determines correlation)
        - `threshold`: Constant value, above which the hypothesis outputs yes, otherwise no
        - Given a vector of inputs, there can be many choices for weight vectors and constant threshold. Such differences expand the vector form into a hypothesis set, where each hypothesis takes the same general form but with different values.

$$\begin{aligned} h(\mathbf{x}) &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \\ &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right) \\ &= \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) \\ &= \text{sign}\left(\mathbf{w}^T \mathbf{x}\right) \end{aligned}$$

2, Perceptron in 2D Space

- Perceptron hypothesis set can occupy any dimensional space

        - $w_0, w_1, ..., w_N$ and $x_1, x_2, ..., x_N$ for $R^N$ space
    - In 2D plane, perceptrons are a set of *lines*, dividing the plane into positive and negative
        - Perceptrons are **linear (binary) classifiers** in 2D space

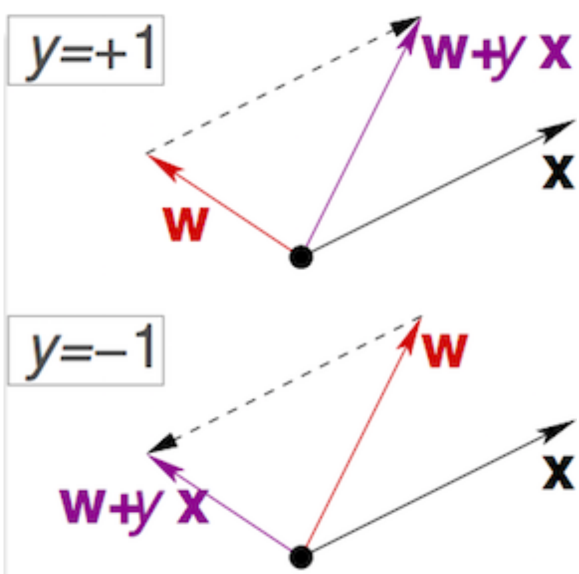



## Perceptron Learning Algorithm (PLA)

1. Selecting hypothesis $g$ from hypothesis set $H$

- Desired $g \approx f$:
    - But the real-world model $f$ is unknown, can only be inferred from sample $D$
    - Hypothesis set $H$ if of infinite size, impossible/inefficient to perform brute-force search
- Idea: Start from some $g_0$, with weight vector $w_0$, and *correct* its deviation from $D$

2. PLA

- For $t = 0, 1, ...$
    - For the current weight vector $w_t$, find a point $(x_{n(t)}, y_{n(t)})$ in $D$ where $sign(w_t^T x_{n(t)}) \neq y_{n(t)}$
        - In other words, where the current model choice has mis-labeled a sample
    - Iterate to the next weight vector $w_{t+1}$, such that $w_t + y_{n(t)} x_{n(t)} \rightarrow w_{t+1}$
        - $w_{t+1}$ is *corrected* by adding cross-product of y and x from the mis-labled sample
        - 

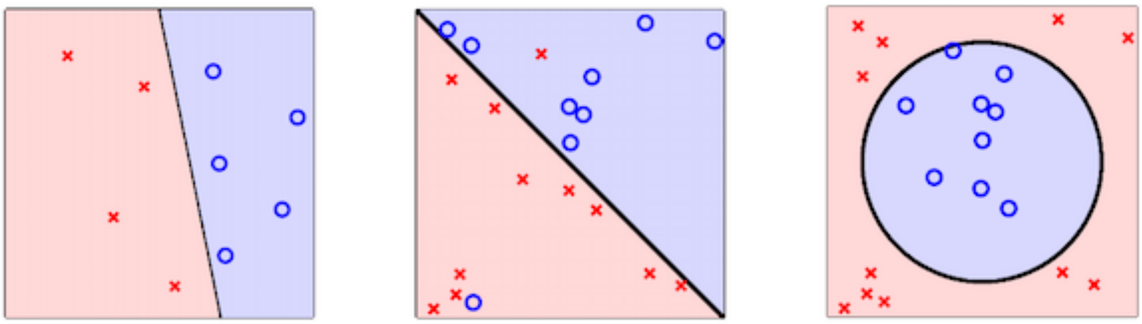
    - Repeat until no more mistakes
    - The last weight vector $w_{PLA}$ is that of the optimal hypothesis $g$

3. Cyclic PLA

- A practical PLA implementation
- Perform the above iterntion is cycles, until no sample is mislabled in a given cycle
- Can follow a naive cycle (1,...,N) or **precomputed random cycle**

### Guarantees of PLA

1. Linear separability

- By definition, the necessary condition for PLA iteration to stop is that, there exists a weight vector $w$ for sample set $D$, such that all points in $D$ can be correctly classified by the hypothesis $g$ based on $w$
- Such $D$ is called **linear separable**
- 

  (linear separable) (not linear separable) (not linear separable)

2. Linear separability implies PLA convergence

- Linear separable $D \Rightarrow$ Exist perfect $w_f$ such that $y_n = sign(w_f^T x_n)$
  - Every data point $x_n$ correctly classified by line $g$
  - $y_{n(t)} w_f^T x_{n(t)} \geq \min_n y_n w_f^T x_n > 0$
- $w_f^T w_t$ *increases* when updated with any previously mis-labeled sample $(x_{n(t)}, y_{n(t)})
  - $$\begin{aligned} \mathbf{w}_f^T \mathbf{w}_{t+1} &= \mathbf{w}_f^T \left( \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)} \right) \\ &\geq \mathbf{w}_f^T \mathbf{w}_t + \min_n y_n \mathbf{w}_f^T \mathbf{x}_n \\ &> \mathbf{w}_f^T \mathbf{w}_t + 0. \end{aligned}$$
  - Given that $w_f^T w_t$ represents the inner product of these two vectors, larger the value, the closer these two vectors are
    - In other words, $w_t \rightarrow w_f^T$ with each update
    - Under the assumption that the length of $w_t$ does not increase steeply with each update (so the inner product growth is mostly due to vectors approaching)
      - Proof:

$$\begin{aligned} \|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)} + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + 0 + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + \max_n \|y_n\mathbf{x}_n\|^2 \end{aligned}$$

- Above holds because by definition of PLA, $w_t$ is updated **only in caase of mistakes**, or:
  - $sign(w_t^T x_{n(t)}) \neq y_{n(t)} \Leftrightarrow y_{n(t)} w_t^T x_{n(t)} \leq 0$

### PLA Convergence Theorem

1. Define $R^2 = \max_n \|x_n\|^2, \rho = \min_n y_n \frac{w_f^T}{\|w_f\|} x_n$, then the perceptron algorithm takes **at most** $\frac{R^2}{\rho^2}$ errors
   - $R^2$ = maximum length of input vector
   - $\rho$ = minimum inner product between target vector from sample, and the unit vector predicted by the PLA line
     - $\rho \geq 0$ because all data points are classified correctly

2. Assumptions:
   * Data set is linear separable
   * Start PLA iteration from $w_0 = 0$

3. Proof:
   - From the section above, we have $\|w_{t+1}\|^2 \leq \|w_t\|^2 + \max_n \|x_n\|^2$
   - Given definition of $R^2$, the above is equivalent to $\|w_{t+1}\|^2 \leq \|w_t\|^2 + R^2$
   - Because PLA starts with $w_0 = 0$, backwards induction t times (to t = 0) results in:
     - $\|w_{t+1}\|^2 \leq TR^2$ (equation 1),
   - In addition, because $w_f^T w_{t+1} \geq w_f^T w_t + \min_n y_n w_f^T x_n$, backwards induction t times gives:
     - $w_{t+1} w_f^T \geq T\rho\|w_f^T\|$
     - Because $\frac{w_{t+1} w_f^T}{\|w_{t+1}\|\|w_f^T\|} \leq 1$ by the definition of inner product, we have:
     - $\|w_{t+1}\| \geq T\rho$ (equation 2)
   - Combining equation 1 and 2 gives:

$$T^2\rho^2 \leq \|w_{T+1}\|^2 \leq TR^2$$

$$\Rightarrow T \leq \frac{R^2}{\rho^2}$$

### Pocket Algorithm

1. PLA assumes data is linear separable, which might not be provable (or simply not true) in real-world use cases
   - Noise
   - Nature of data
2. Instead, try to find the weights that **minimize** errors on input data

$$w_g \leftarrow \arg\min_{w} \sum_{n=1}^{N} [y_n \neq sign(w^T x_n)]$$

3. Pocket algorithm: Iteratively, keep the current best weights in pocket

initialize pocket weights $\hat{\mathbf{w}}$

For $t = 0, 1, \cdots$

1. find a (random) mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$
2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

3. if $\mathbf{w}_{t+1}$ makes fewer mistakes than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by $\mathbf{w}_{t+1}$

...until enough iterations

return $\hat{\mathbf{w}}$ (called $\mathbf{w}_{\text{POCKET}}$) as $g$