# Machine Learning Principles

Class9 : Oct. 2

Linear Classification III: Perceptron

Instructor: Diana Kim

# Today's Lecture

1. Perceptron modeling (Rosenblatt, 1962)
2. Training: Defining an Objective Function and Optimization
3. *Convergence Theorem
4. Comparison to Logistic Regression
5. Perceptron as a Foundational Elements of Neural Nets
6. Perceptron for Logic gates

- Perceptron

## ** in previous class,

In the last two classes,
we studied how the two discriminant functions for binary classifications can be designed by learning $P[C_0 \mid x]$ & $P[C_1 \mid x]$

- Generative: GDA, Naïve Bayes
- Discriminative: Logistic Regression

## ** in previous class,

- binary decision rule

$$f_0(x) \underset{H_1}{\overset{H_0}{\gtreqless}} f_1(x)$$

- the decision boundary was can be hyperplane $\longrightarrow$ $\boxed{\vec{w}^t \Phi(x) = 0}$

$\Phi(x)$ is a feature set.

## ** in previous class,

Example) suppose we learned two functions over $2D$ space $(x, y)$. Compute a hyperplane that defines the decision boundary.

$$\overbrace{2x + y - 1}^{f_0(x)} \underset{H_1}{\overset{H_0}{\gtrless}} \overbrace{x + y + 1}^{f_1(x)}$$

- A binary classification problem can be finding a hyperplane.

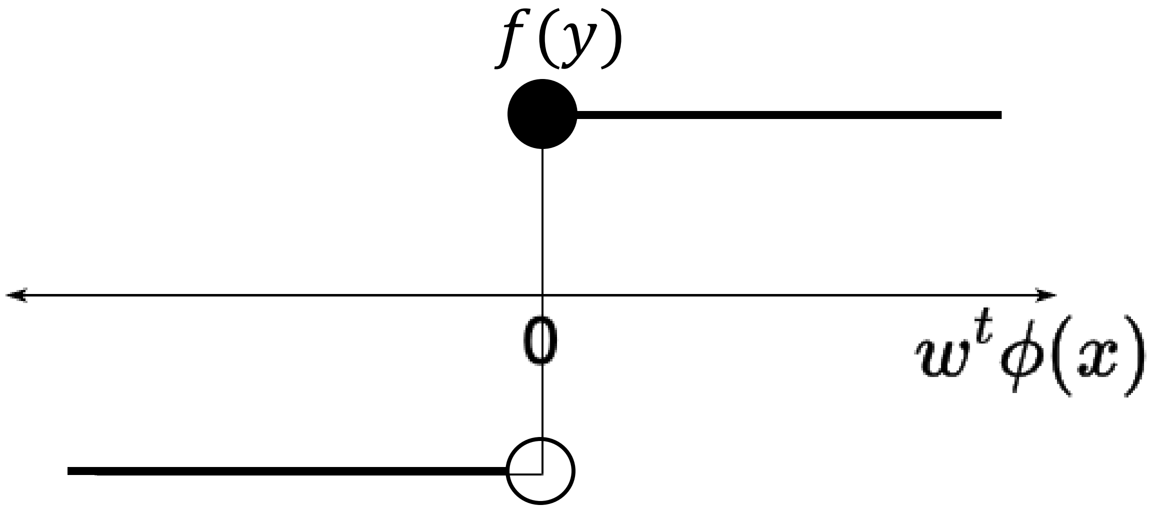
# [1] Perceptron (learning decision boundary)

Today we are going to study **Perceptron Algorithm** [Rosenblatt (1962)]
It directly learns a decision boundary (hyperplane) for binary classification without considering any probabilistic modeling.

$$\vec{w}^t \Phi(x) \underset{H_0}{\overset{H_1}{\gtrless}} 0$$

# [2] Perceptron (activation function: step function)

In perceptron modeling
step activation function is used to predict outcome class directly.

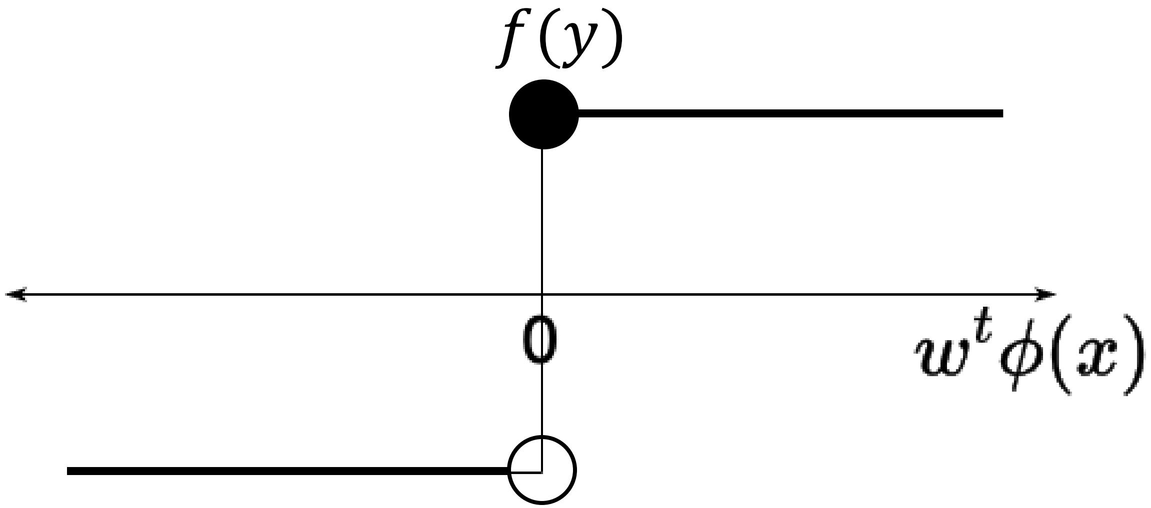



$$\begin{cases} f(y) = 1 & y = \vec{w}^t \phi(x) \geq 0 \\ f(y) = -1 & y = \vec{w}^t \phi(x) < 0 \end{cases}$$

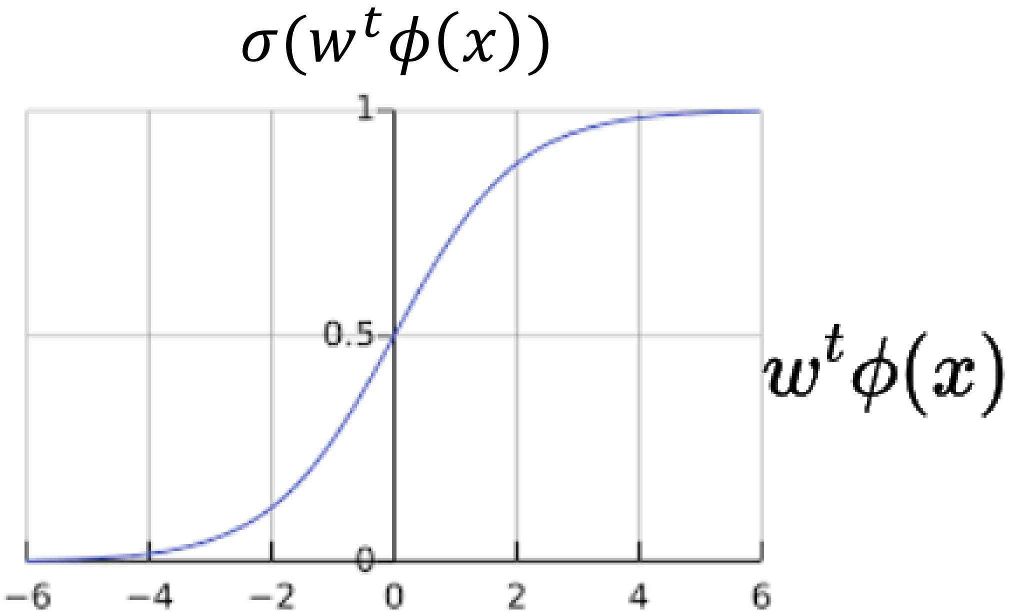
- If right classification, the product between prediction $\vec{w}\phi(x)$ and ground gruth (t) is positive $\vec{w}\phi(x) \cdot t \geq 0$

# [3] Perceptron (perceptron vs. logistic regression)

- Perceptron outcome does not give probabilistic interpretation




[perceptron activation]
: <u>hard</u> decision (non-probabilistic)




[Logistic Regression activation]
: <u>soft</u> decision (probabilistic)

- Training Perceptron

  (no MLE but minimizing “misclassification” error)

# [1] Training (objective function)

- misclassification error

$$E(\vec{w}) = -\sum_{n \in \mathcal{M}} \vec{w}^t \phi(x_n) \cdot t_n - \sum_{n \in \mathcal{M}^c} \cdot 0$$

$\mathcal{M}$ : misclassification samples
(in fact, this is a sample)

- the objective function is piecewise linear (how), but differentiable at the current *w*.

$$\nabla E(\vec{w}) = -\sum_{n \in \mathcal{M}} t_n \phi(x_n)$$

# [2] Training (updating rule by Rosenblatt, 1962)

- perceptron learning rule (on-line fashion):

$$w(t+1) = w(t) - \eta \nabla E(w) = w(t) + (\eta = 1) \cdot \phi(\vec{x_n}) t_n$$

- update current *w* by one misclassified sample. (one by one)

Q: can we guarantee that it decreases the overall classification error?

# [2] Training (updating rule by Rosenblatt, 1962)

- after one sample update,
  the contribution to the error from a misclassified pattern will be reduced.

$$\underline{-w_{\tau+1}^t\phi(x_n)t_n} = -\{w_\tau + \phi(x_n)t_n\}^t\phi(x_n)t_n = -w_\tau^t\phi(x_n)t_n - ||\phi(x_n)t_n||^2 < \underline{-w_\tau^t\phi(x_n)t_n}$$

- however, (1) this does not mean that the contribution from the other misclassification will be reduced. (2) furthermore, the update may cause come right classified patterns to become misclassification.

# [3] Training (updating rule by Rosenblatt, 1962)

- perceptron algorithm is not guaranteed to reduce the total error function at each iteration.

# [4] Training (Convergence Theorem)

Even though each iteration does not guarantee that the algorithm moving to a descent direction of misclassification error,
Rosenblatt proved that if training data set is linearly separable,
then the perceptron algorithm is guaranteed to find a solution in finite number of iterations.

- Convergence Theorem, Perceptron

if training data set is linearly separable,
then the perceptron algorithm is guaranteed to find a solution in finite number of iterations.

# [1] Convergence Theorem

if training data set is linearly separable,
then the perceptron algorithm is guaranteed to find a solution in finite number of iterations.

$$\text{if} \quad \exists w_* \quad \text{such that } y_i \cdot w_*^t x_i > 0 \quad \forall i$$

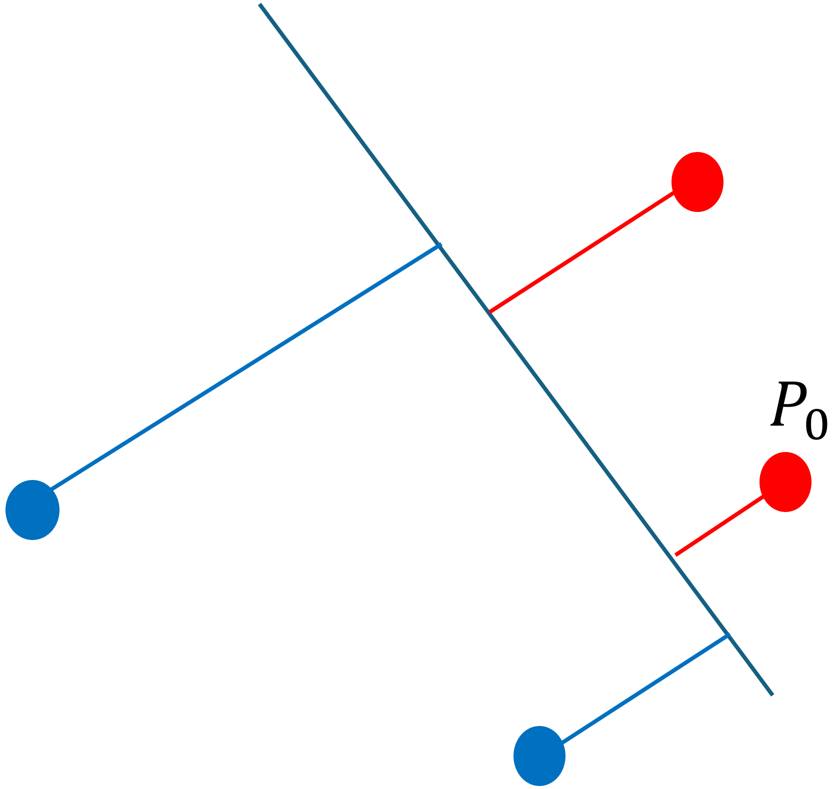
[linear separable]

$$\text{then } \# \text{ of iterations} \le C$$

[convergence within finite iterations]

# **Concept of Margin (preliminary)

- margin: the smallest distance
  between the decision boundary and any of the samples

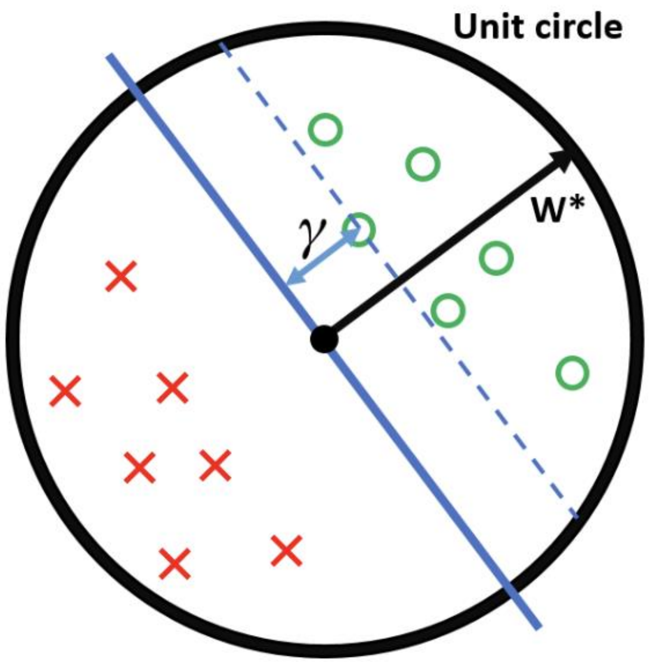



- the distance (d) of $p_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ to
  the hyperplane $w^t \begin{bmatrix} x \\ y \end{bmatrix} + b$ is

$$d = \frac{|w^t \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + b|}{||w||}$$

# [2] Convergence Theorem (proof)

- suppose
  - $\exists w_* \quad \text{such that } w_*^t x_i \cdot y_i > 0 \quad \forall i$
  - $||w*|| = 1$
  - $||x|| \leq 1$
  - margin: $\gamma$




[$w^*$ works at the space scaled by "s"]

**

$$t_i \cdot w*^t x_i \geq 0 \quad \forall i \quad \leftrightarrow \quad t_i \cdot w*^t (x_i/s) \geq 0 \quad \forall i \quad \text{and} \quad s > 0$$

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

# [3] Convergence Theorem (proof)

- suppose $(\vec{x}, y)$ is a misclassification sample by $w_o \rightarrow y\, {w_o}^t x < 0$
- one update satisfy below;

$$w_1 = w_0 + \eta \cdot x \cdot y$$

$$\boxed{w_1^t w_*} = (w_0 + \eta \cdot x \cdot y)^t w_* = w_0^t w_* + \underline{\eta y \cdot x^t w_*} \geq w_0^t w_* + \eta\gamma$$

**two inner product the distance of $(x, y)$ to the hyperplane

$$\boxed{w_1^t w_1} = (w_0 + \eta \cdot x \cdot y)^t \cdot (w_0 + \eta \cdot x \cdot y) = w_0^t w_0 + 2 \cdot \eta \cdot \underline{y \cdot w_0^t \cdot x} + \eta^2 y^2 x^t x \leq w_0^t w_0 + \eta^2$$

$< 0, (x, y)$ is the misclassification sample

# [4] Convergence Theorem (proof)

- any $M$ updates, satisfy below

$$w_M^t w_* \geq w_0^t w_* + M\eta\gamma$$

$$w_M^t w_M \leq w_0^t w_0 + M\eta^2 = ||w_0||^2 + M\eta^2$$

- any $M$ update is bounded by 1/margin$^2$

  (for simplicity assuming $\overrightarrow{w_0} = 0$)

$$\begin{aligned} M\eta\gamma &\leq w_M^t w_* \\ &\leq ||w_M|| \cos\theta \\ &\leq ||w_M|| \leq \sqrt{M}\eta \\ M\gamma &\leq \sqrt{M} \\ M &\leq \frac{1}{\gamma^2} \end{aligned}$$

# [5] Convergence Theorem (proof)

The proof showed that
when there exist a hyperplane linearly separates the data,
the perceptron algorithm converges regardless of initial parameter $w_0$ and step size η.

- Perceptron, steps

# [1] Training (Perceptron Learning Algorithm by Rosenblatt in 1962)

**Perceptron Algorithm**

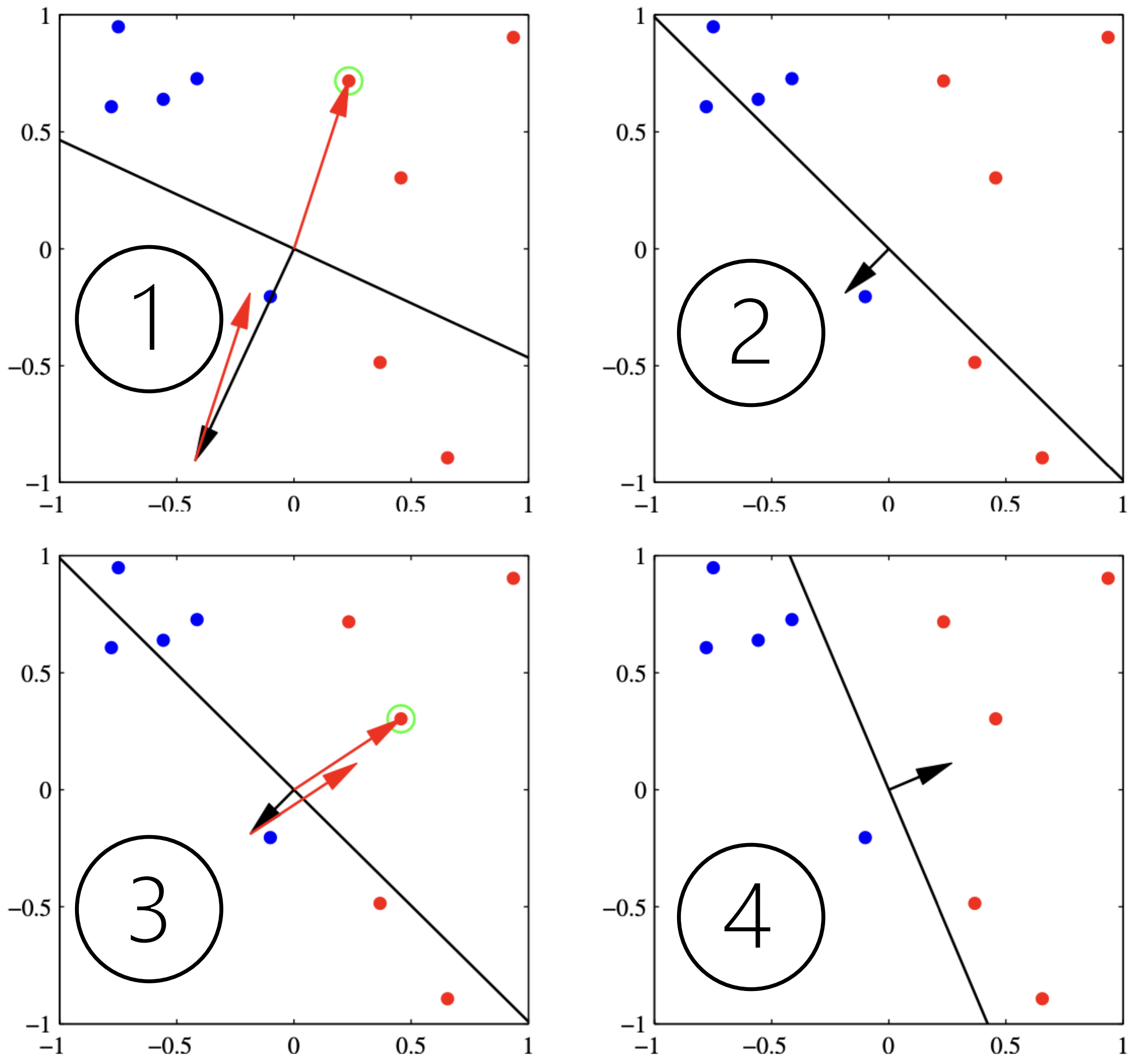
Initialize $\vec{w} = \vec{0}$
**while** TRUE **do**
  $m = 0$
  **for** $(x_i, y_i) \in D$ **do**
    **if** $y_i(\vec{w}^T \cdot \vec{x}_i) \leq 0$ **then**
      $\boxed{\vec{w} \leftarrow \vec{w} + y\vec{x}}$ update one sample by one!
      $m \leftarrow m + 1$
    **end if**
  **end for**
  **if** $m = 0$ **then**
    break
  **end if**
**end while**

Source from: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

# [2] Training (textbook figure)

$$w(t+1) = w(t) - \eta \nabla E(w) = w(t) + (\eta = 1) \cdot \phi(\vec{x_n}) t_n$$

suppose red : $+1$
blue : $-1$




The Convergence of Perceptron
from Text Bishop Figure 4.7

- Comparison between

  Logistic Regression vs. Perceptron

# [1] Logistic Sigmoid Regression vs Perceptron

1. Both are the algorithm for binary classification. (T/F)

2. When data is not linearly separable, then both does not converge. (T/F)

3. When data is linearly separable, logistic regression curve would resemble the Perceptron's sign function. (T/F)

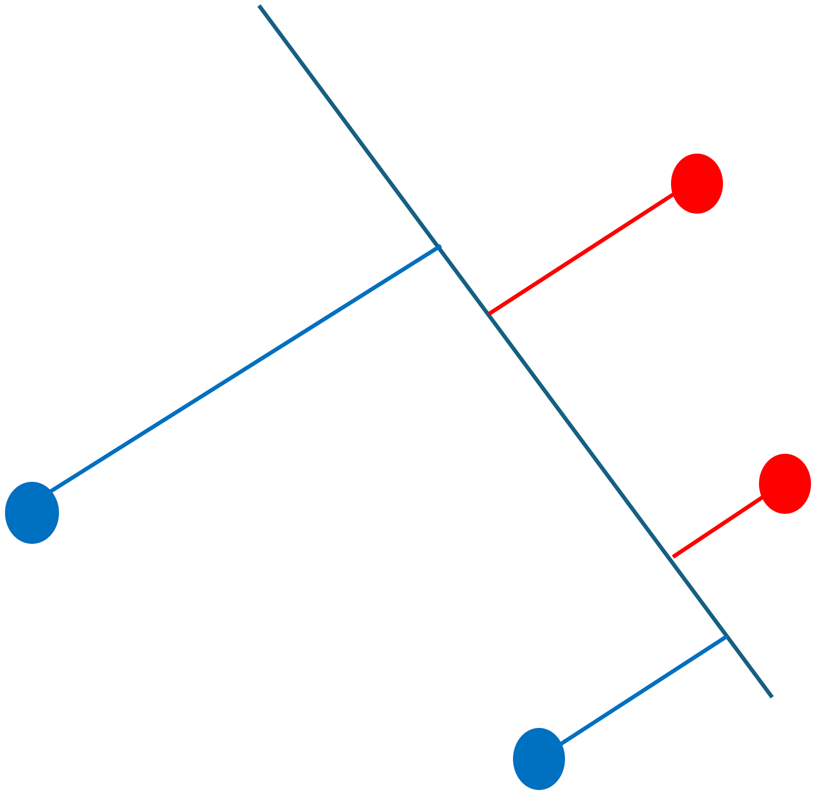
4. Only Perceptron defines a decision boundary. (T/F)

5. Perceptron promotes a large margin classifier. (T/F)

6. Logistic regression promotes a large margin classifier. (T/F)

## **Concept of Margin

- margin: the smallest distance
  between the decision boundary and any of the samples



## [2] Logistic Sigmoid Regression vs. Perceptron (loss for a single point $\{x, t\}$)

$$\text{Perceptron Loss } (x,t) = \begin{cases} -w^t x \cdot t & w^t x \cdot t < 0 \\ 0 & w^t x \cdot t \geq 0 \end{cases}$$

[misclassification]

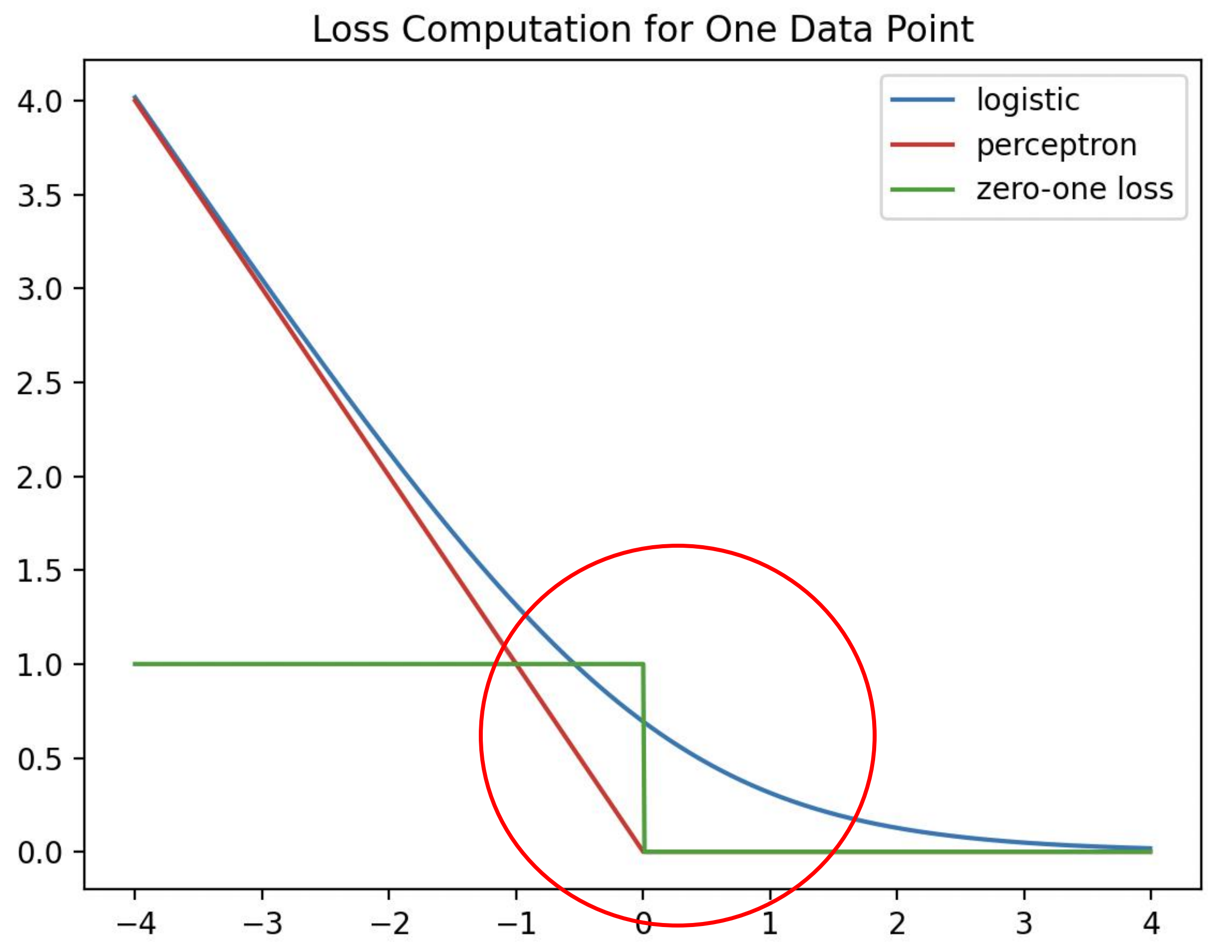
[right classification]

$$\text{Logistic Loss } (x,t) = \begin{cases} -\ln \sigma(w^t x) = -\ln \dfrac{1}{1+\exp(-w^t x)} & t = +1 \\ -\ln \sigma(-w^t x) = -\ln \dfrac{\exp(-w^t x)}{1+\exp(-w^t x)} & t = -1 \end{cases}$$

$$\text{Logistic Loss } (x,t) = -\ln \sigma(w^t x \cdot t) = -\ln \frac{1}{1+\exp(-w^t x \cdot t)}$$

graph for
$X = w^t \cdot x \cdot t$

# [3] Logistic Sigmoid Regression vs Perceptron (loss comparison)




- perceptron does not penalize a small margin while logistic promotes a large margin.

- for $yw^tx < 0$ (misclassification), the perceptron and logistic loss behavior is asymptotically similar.

- Perceptron as a foundational element of neural nets

# [1] Perceptron as foundational an element of neural nets

Perceptron is a building block for the neural net but had not had attention in spite of the convergence theorem. In the book “Perceptron”, Minky and Papert showed that executing certain common computations on Perceptron would be impractically time consuming.

(1) data separability
(2) no way to see high dimensional data is separable or not until running the algorithm
(3) no automatic way to learn feature making the linearly separable

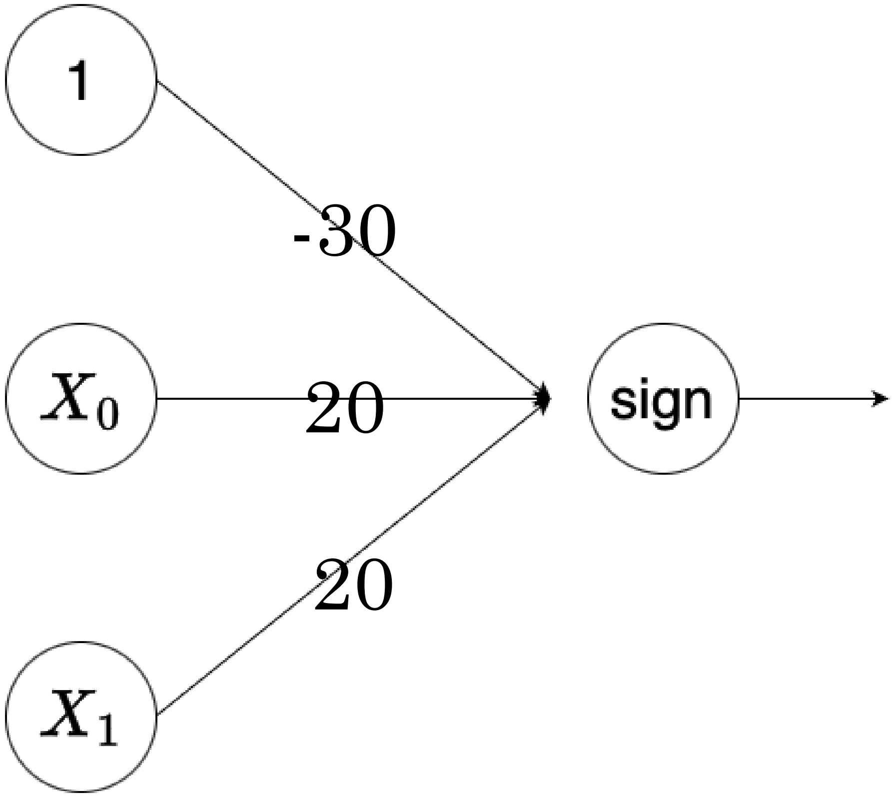
# [2] Perceptron as foundational an element of neural nets

The structure of Perceptron started to work right as having multiple layers. By the 1980s, researchers developed algorithms (backpropagation and different loss) for training with more than one layer, removing many of the limitations identified by Minsky. However, for the limitations by computational power, the # layers were two / three layers. Very recently for modern GPU, very long neural net are being developing. (using trillions parameters in GPT-4, believed to be)

From MIT news article *"Explained: Neural networks Ballyhooed artificial-intelligence technique Known as "deep learning" revives 70-year-old idea."*

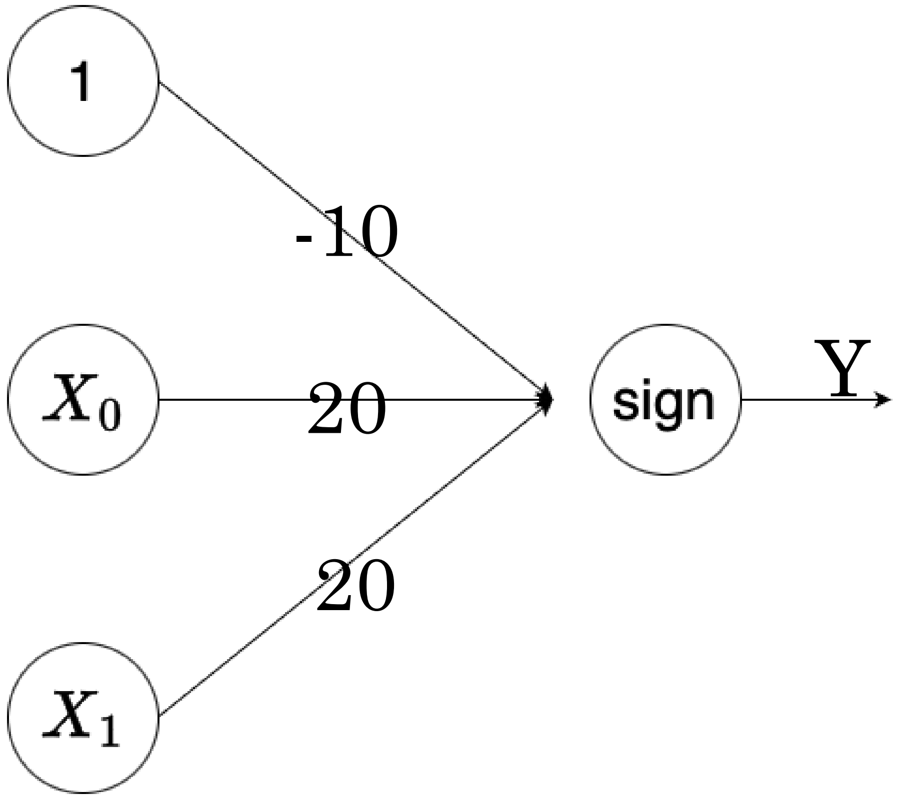
- Solving XOR problem using Perceptron

# [1] Solving XOR problem using Perceptron (AND gate)



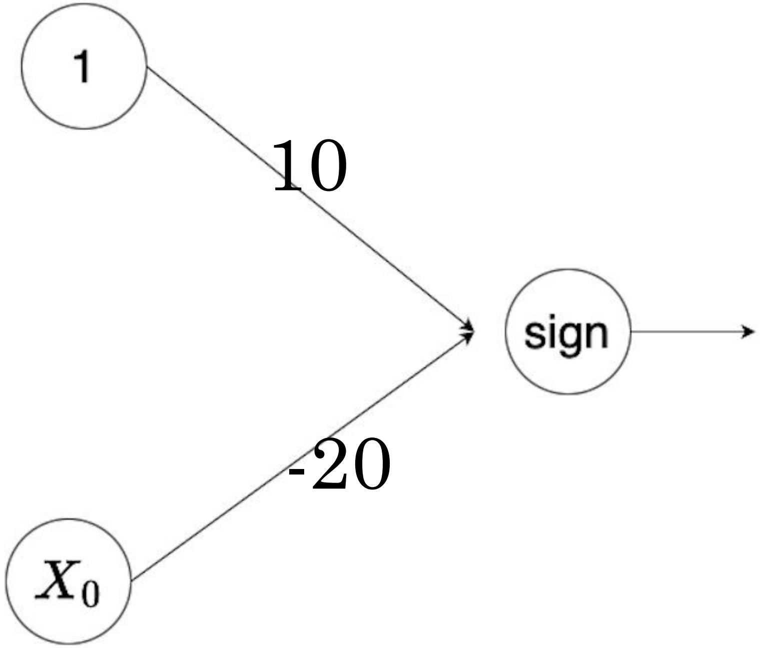

| $X_0$ | $X_1$ | $X_0 \wedge X_1$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

# [2] Solving XOR problem using Perceptron (OR gate)



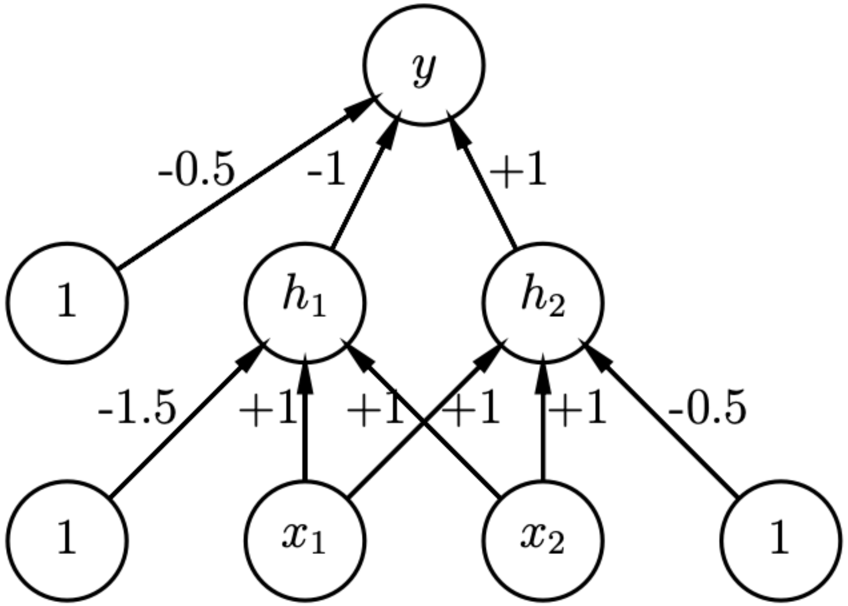

| $X_0$ | $X_1$ | $X_0 \vee X_1$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

# [3] Solving XOR problem using Perceptron (Negation)




| $X_0$ | $\sim X_0$ |
|---|---|
| 0 | |
| 1 | |

# [4] Solving XOR problem using Perceptron (XOR)




$\overline{X_1 \wedge X_2} \wedge (X_1 \vee X_2) = X_1 \oplus X_2$

| $X_1$ | $X_2$ | $X_1 \oplus X_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |