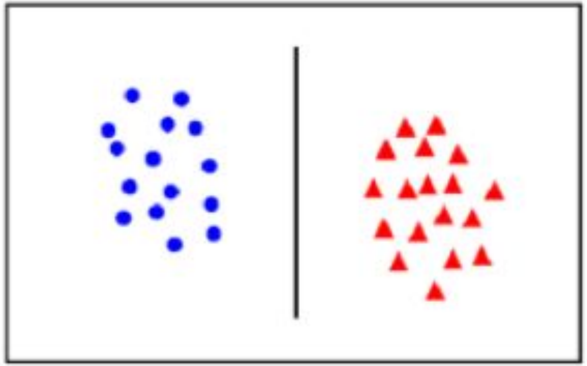
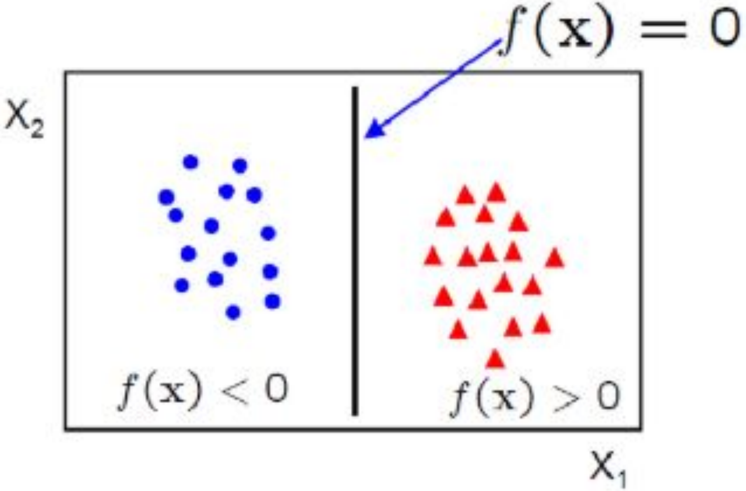
Perceptron

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

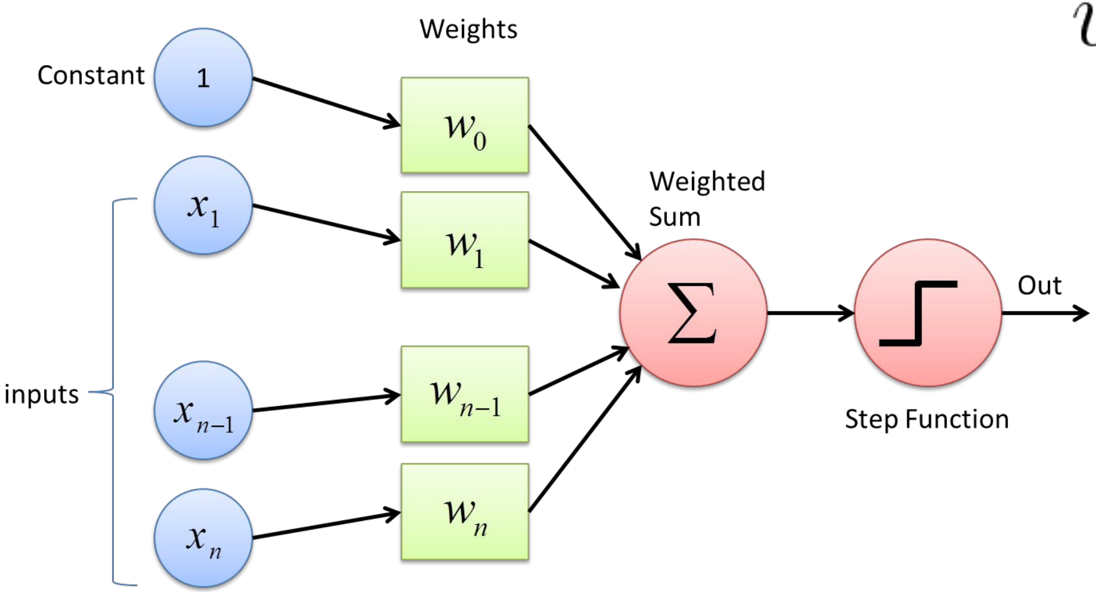
Perceptron

- **Input: D** = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ s.t., $\mathbf{x_i} \in \mathbf{X}, \mathbf{y_i} \in \mathbf{Y}$ where $X \in R$, $Y \in \{-1,1\}$
- Use a function to map real numbers to {-1,1}





Perceptron: Binary Classification



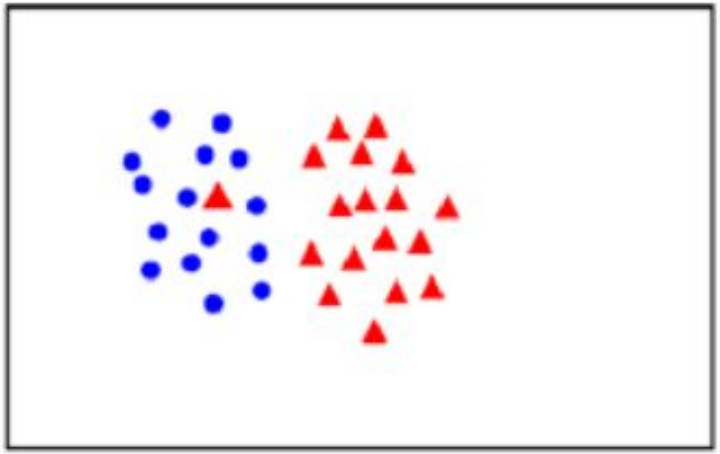
$$v = \sum_{i=1}^{m} w_i x_i + b$$

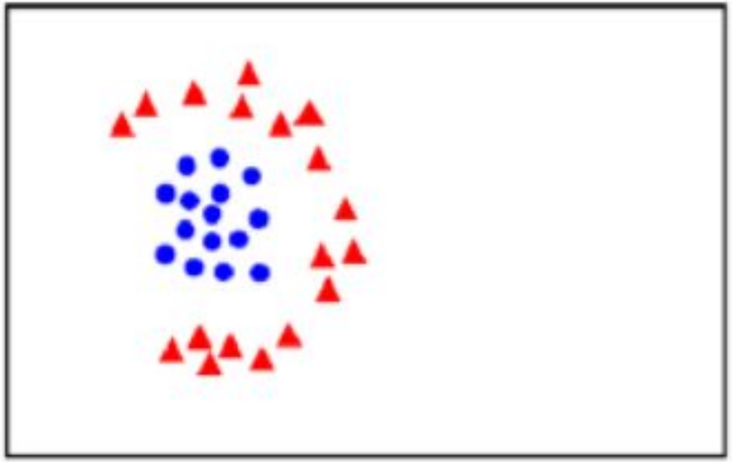
$$\sum_{i=1}^{m} w_i x_i + b = 0$$

$$sgn(v) = \begin{cases} +1 \text{ if } v \geq 0, \\ -1 \text{ otherwise.} \end{cases}$$

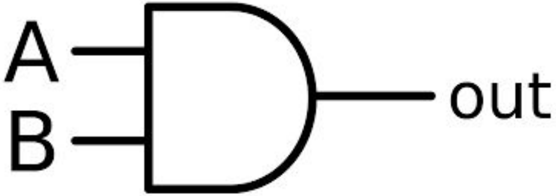
Separability: Linear vs Non-Linear

- **Assumption:** There exists a hyperplane!

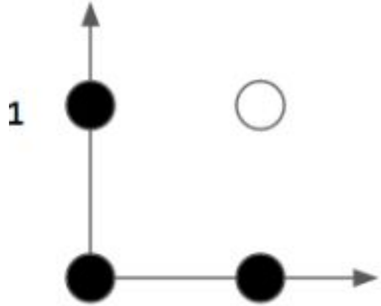




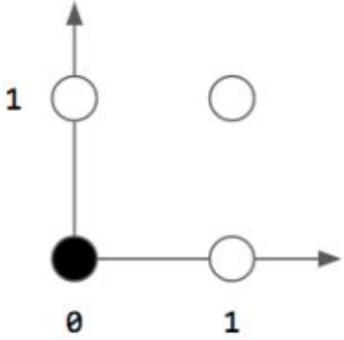
Logical Operations



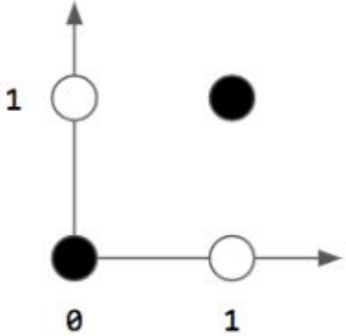
AND



OR



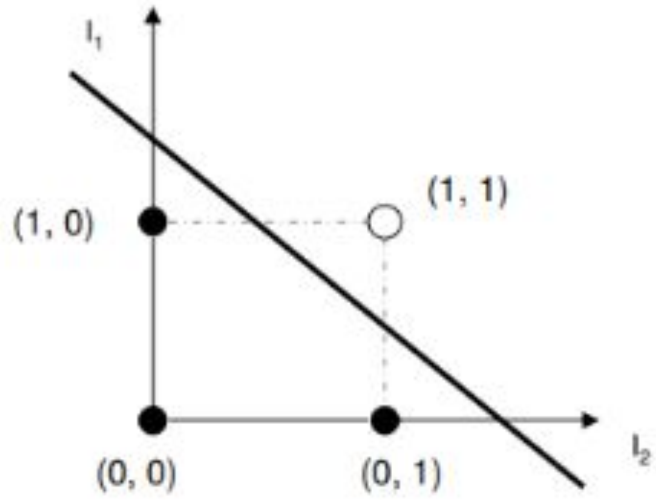
XOR



Example: AND Gate

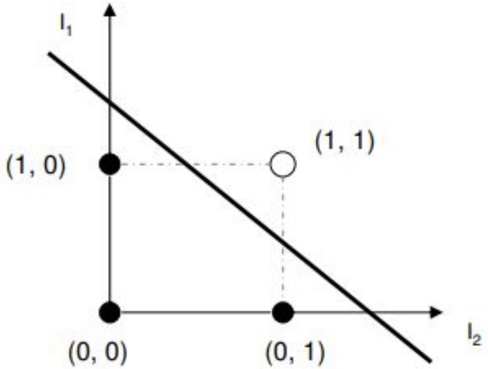
$$x_1 + x_2 - 1.5 = 0$$
$$w_1 = w_2 = 1; b = -1.5$$

1. $x_1 = 0, x_2 = 0$
   a. $0 + 0 - 1.5 = -1.5$
2. $x_1 = 0, x_2 = 1$
   a. $0 + 1 - 1.5 = -0.5$
3. $x_1 = 1, x_2 = 0$
   a. $1 + 0 -1.5 = -0.5$
4. $x_1 = 1, x_2 = 1$
   a. $1 + 1 -1.5 = 0.5$

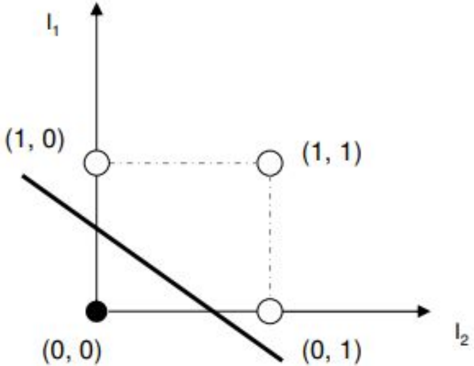


Hyperplane for Logical Operations

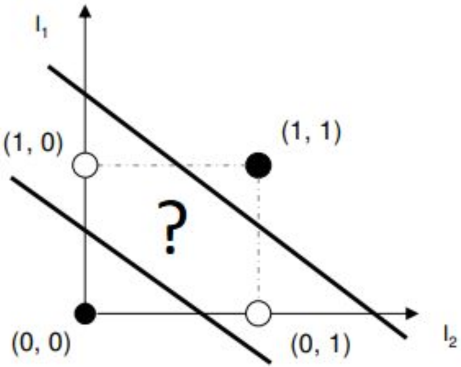
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



Learning

- Correct Classification: $y_i f(x_i) > 0$
  - $f(x_i) = b + w_1x_1 + w_2x_2 + \ldots. w_nx_n > 0$ and belongs to C_1
  - $f(x_i) = b + w_1x_1 + w_2x_2 + \ldots. w_nx_n < 0$ and belongs to C_2
  - $w(i+1) = w(i)$

- Incorrect Classification: $y_i f(x_i) < 0$
  - $f(x_i) = b + w_1x_1 + w_2x_2 + \ldots. w_nx_n < 0$ and belongs to C_1
    - $w(i+1) = w(i) + \Delta$
  - $f(x_i) = b + w_1x_1 + w_2x_2 + \ldots. w_nx_n > 0$ and belongs to C_2
    - $w(i+1) = w(i) - \Delta$

Perceptron Learning Algorithm

- **Input:** Training examples $\{x_i, y_i\}_{i = 1 \text{ to } n}$
- Initialize w and b as zero or randomly
- While !converged do
  - #Loop through the samples
  - *for j = 1 to n do*
    - *#Compare the true label and the prediction*
    - $error_j = y_j - \varphi(w^T x_j + b)$
    - *#If the model wrongly predicts the class, update the weight and the bias*
    - *If error != 0*
      - *#Update the weight*
        - $W = w + error_j \times x_j$
      - *#Update the bias*
        - $B = b + error_j$
  - *Test for convergence*
- **Output:** Set of weights **w** and bias **b** for the perceptron

Note: Φ = sgn function

Perceptron Learning Algorithm

Let’s follow the steps manually for the AND gate.

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

AND		
I_1	I_2	out
0	0	~~0~~
0	1	~~0~~
1	0	~~0~~
1	1	1

0 -> -1
1 -> 1

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - W_1 ,W_2 = 0 and b = 0
  - *Let's calculate error and compare the true label and the prediction*
    - *sample1_prediction = 1*
    - *sample1_error = -1 - 1 = -2*

Iteration=1

AND			
I_1	I_2	out	
0	0	0	**-1**
0	1	0	**-1**
1	0	0	**-1**
1	1	1	

***error** = $y - \varphi(w^T x + b)$*

Φ = sgn function

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  Iteration=1
  - W_1, W_2 = 0 and b = 0
  - *Let's calculate error and compare the true label and the prediction*
    - *sample1_prediction = 1*
    - *sample1_error = -1 - 1 = -2*
  - *If error != 0*
    - *Updating the weights*
      - *W1 = 0 + (-2)*0 = 0*
      - *W2 = 0 + (-2)*0 = 0*
    - *Updating the bias*
      - *B = 0 + (-2) = -2*
  - W_1, W_2 = 0 and b = -2

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

$$\boldsymbol{error = y - \varphi(w^T x + b)}$$
$$\boldsymbol{W = w + error \times x}$$
$$\boldsymbol{B = b + error}$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - $W_1, W_2 = 0$ and $b = -2$
  - *Let's calculate error and compare the true label and the prediction*
    - *sample2_prediction = -1*
    - *sample2_error = -1 + 1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - $W_1, W_2 = 0$ and $b = -2$

Iteration=1

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

-1
-1
-1

$$\textbf{error} = y - \varphi(w^T x + b)$$
$$W = w + \textbf{error} \times x$$
$$B = b + \textbf{error}$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - W_1 ,W_2 = 0 and b = -2
  - *Let's calculate error and compare the true label and the prediction*
    - *sample3_prediction = -1*
    - *sample3_error = -1 + 1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - W_1 ,W_2 = 0 and b = -2

Iteration=1

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

error = y - φ(w^T x + b)
W = w + error ✖ x
B = b + error

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

AND			
I_1	I_2	out	
0	0	~~0~~	-1
0	1	~~0~~	-1
1	0	~~0~~	-1
1	1	1	

- Initialize w and b as zero.

Iteration=1

  - W_1, W_2 = 0 and b = -2
  - *Let's calculate error and compare the true label and the prediction*
    - *sample4_prediction = -1*
    - *sample4_error = 1 + 1 = 2*
  - *If error != 0*
    - *Updating the weights*
      - *W1 = 0 + (2)*1 = 2*
      - *W2 = 0 + (2)*1 = 2*
    - *Updating the bias*
      - *B = -2 + (2) = 0*
  - W_1, W_2 = 2 and b = 0

error = y - φ(w^T x + b)
W = w + error ✖ x
B = b + error

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.

Iteration=2

  - W_1 ,W_2 = 2 and b = 0
  - *Let's calculate error and compare the true label and the prediction*
    - *sample1_prediction = 1*
    - *sample1_error = -1 -1 = -2*
  - *If error != 0*
    - *Updating the weights*
      - *W1 = 2 + (-2)*0 = 2*
      - *W2 = 2 + (-2)*0 = 2*
    - *Updating the bias*
      - *B = 0 + (-2) = -2*
  - W_1 ,W_2 = 2 and b = -2

AND			
I_1	I_2	out	
0	0	~~0~~	-1
0	1	~~0~~	-1
1	0	~~0~~	-1
1	1	1	

$$error = y - \varphi(w^T x + b)$$
$$W = w + error ✖ x$$
$$B = b + error$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - W_1 ,W_2 = 2 and b = -2
  - *Let's calculate error and compare the true label and the prediction*
    - *sample2_prediction = 1*
    - *sample2_error = -1 -1 = -2*
  - *If error != 0*
    - *Updating the weights*
      - *W1 = 2 + (-2)*0 = 2*
      - *W2 = 2 + (-2)*0 = 2*
    - *Updating the bias*
      - *B = -2 + (-2) = -4*
  - W_1 ,W_2 = 2 and b = -4

Iteration=2

Error is made intentionally (try correcting it)

AND			
I_1	I_2	out	
0	0	~~0~~	-1
0	1	~~0~~	-1
1	0	~~0~~	-1
1	1	1	

error = y - φ(w^T x + b)
W = w + error ✖ x
B = b + error

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1
- Initialize w and b as zero.
  - W_1 ,W_2 = 2 and b = -4
  - *Let's calculate error and compare the true label and the prediction*
    - *sample3_prediction = -1*
    - *sample3_error = -1 +1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - W_1 ,W_2 = 2 and b = -4

Iteration=2

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

error = y - φ(w^T x + b)
W = w + error ✖ x
B = b + error

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - W_1 ,W_2 = 2 and b = -4
  - *Let's calculate error and compare the true label and the prediction*
    - *sample4_prediction = 1*
    - *sample4_error = 1 -1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - W_1 ,W_2 = 2 and b = -4

Iteration=2

AND			
I_1	I_2	out	
0	0	~~0~~	-1
0	1	~~0~~	-1
1	0	~~0~~	-1
1	1	1	

$$error = y - \varphi(w^T x + b)$$
$$W = w + error ✖ x$$
$$B = b + error$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.
  - $W_1, W_2 = 2$ and $b = -4$
  - *Let's calculate error and compare the true label and the prediction*
    - *sample1_prediction = -1*
    - *sample1_error = -1 +1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - $W_1, W_2 = 2$ and $b = -4$

Iteration=3

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

$$error = y - \varphi(w^T x + b)$$
$$W = w + error \times x$$
$$B = b + error$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1
- Initialize w and b as zero.
  - $W_1, W_2 = 2$ and $b = -4$
  - *Let's calculate error and compare the true label and the prediction*
    - *sample2_prediction = -1*
    - *sample2_error = -1 +1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - $W_1, W_2 = 2$ and $b = -4$

Iteration=3

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

error = y - φ(w^{T}x + b)
W = w + error ✖ x
B = b + error

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

AND		
I_1	I_2	out
0	0	~~0~~ -1
0	1	~~0~~ -1
1	0	~~0~~ -1
1	1	1

- Initialize w and b as zero.

Iteration=3

  - $W_1, W_2 = 2$ and $b = -4$
  - *Let's calculate error and compare the true label and the prediction*
    - *sample3_prediction = -1*
    - *sample3_error = -1 +1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - $W_1, W_2 = 2$ and $b = -4$

$$error = y - \varphi(w^T x + b)$$
$$W = w + error \times x$$
$$B = b + error$$

Perceptron Learning Algorithm

- **Input:** Training examples for the AND gate problem?
  - Input is clear.
  - Map the out to -1 and 1 s.t., 1 maps to 1 and 0 maps to -1

- Initialize w and b as zero.

Iteration=3

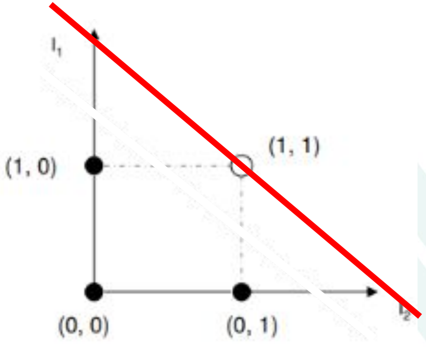
  - W_1 ,W_2 = 2 and b = -4
  - *Let's calculate error and compare the true label and the prediction*
    - *sample4_prediction = 1*
    - *sample4_error = 1 -1 = 0*
  - *If error != 0*
    - *Updating the weights*
    - *Updating the bias*
  - W_1 ,W_2 = 2 and b = -4

No change in iteration 3 i.e, error was 0 for all the training examples. The model has converged.

- **Output:** W_1 ,W_2 = 0.5 and b = -1

AND			
I_1	I_2	out	
0	0	0	-1
0	1	0	-1
1	0	0	-1
1	1	1	

error = y - φ(w^T x + b)
W = w + error ✖ x
B = b + error



Perceptron Convergence Theorem

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T\mathbf{x}(n) \le 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) \le 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

Assumptions:

- Learning rate $\eta = 1$
- Initial Condition $w(0) = 0$

Perceptron Convergence Theorem

- Misclassification for x(1), x(2)..., x(n) ε C_1
  - *w(n+1)*
    - $= w(n) + x(n)$
  - *w(0) = 0;*
    - $w(1) = x(0)$
    - $w(2) = w(1) + x(1)$
  - $w(n+1) = x(0) + x(1) + x(2) + \ldots. + x(n)$ ---- *(I)*
- Since C_1 and C_2 are linearly separable, there exists optimal $\mathbf{w_o}$ such that $w_o^T x(n) > 0$ for $x(1), x(2)\ldots, x(n)\ \varepsilon\ C1$
  - *Let α be a positive number*
    - $$\alpha = \min_{x(n)\varepsilon C_1} w_o^T x(n)$$

Perceptron Convergence Theorem

- Multiplying w_o^T both the sides of the equation (I)
  - $w_o^T w(n+1) = w_o^T x(1) + w_o^T x(2) + ... + w_o^T x(n)$
  - $w_o^T w(n+1) >= n\alpha$
- *Cauchy-Schwarz inequality for two vectors u, v*
  - $||u||^2||v||^2 >= [uv]^2$
    - $||w_o^T||^2||w(n+1)||^2 >= [w_o^T w(n+1)]^2$
- $||w_o^T||^2||w(n+1)||^2 >= n^2\alpha^2$
  - $||w(n+1)||^2 >= n^2\alpha^2/||w_o^T||^2$ .... (II)

Perceptron Convergence Theorem

- $w(k+1) = w(k) + x(k), k = 1, 2, \ldots n$ *and* $x(k)\ \varepsilon\ C_1$
- By taking the squared Euclidean norm
  - $||w(k+1)||^2 = ||w(k)||^2 + ||x(k)||^2 + 2w^T(k)x(k)$
- Misclassification for x(1), x(2)..., x(n) $\varepsilon\ C_1$ i.e. $w^T(k)x(k) < 0$
  - $||w(k+1)||^2 <= ||w(k)||^2 + ||x(k)||^2$
- $||w(k+1)||^2 - ||w(k)||^2 <= ||x(k)||^2$ for k =1, 2, ...n
- w(0) = 0

$$\|w(n+1)\|^2 \leq \sum_{i=1}^{n} \|x(k)\|^2$$

$$\|w(n+1)\|^2 \leq n\beta \qquad \beta = \max_{x(n)\varepsilon C_1} \|x(k)\|^2$$

Perceptron Convergence Theorem

- Conflict

$$\|\mathbf{w}(n+1)\|^2 \geq \frac{n^2\alpha^2}{\|\mathbf{w}_0\|^2}$$

$$\|\mathbf{w}(n+1)\|^2 \leq \sum_{k=1}^{n} \|\mathbf{x}(k)\|^2 \leq n\beta$$

- n cannot be larger than some value n_{max} for which both are satisfied:
  - $$\frac{n_{\max}^2\alpha^2}{\|\mathbf{w}_0\|^2} = n_{\max}\beta \qquad n_{\max} = \frac{\beta\|\mathbf{w}_0\|^2}{\alpha^2}$$

Perceptron Convergence Theorem

- Weight update algorithm must terminate after n_{max} iterations.
  - If the data are linearly separable, perceptron is guaranteed to converge
- There is no unique solution for w_o (optimal weights) and n_{max} (maximum number of iterations).
- Fixed Increment Convergence Theorem:
  - For some $n_o <= n_{max}$, the perceptron converges such that $w(n_o) = w(n_o + 1) = w(n_o + 2) = ...$
    - *n_o: optimal number of iterations*

Loss: Perceptron and Logistic Regression

Φ = sgn function

$$\mathcal{L}_{\text{perc}}(x, y) = \begin{cases} 0 & \text{if } \; y\Phi(w^T x) > 0 \\ -y\Phi(w^T x) & \text{if } \; y\Phi(w^T x) \leq 0 \end{cases}$$

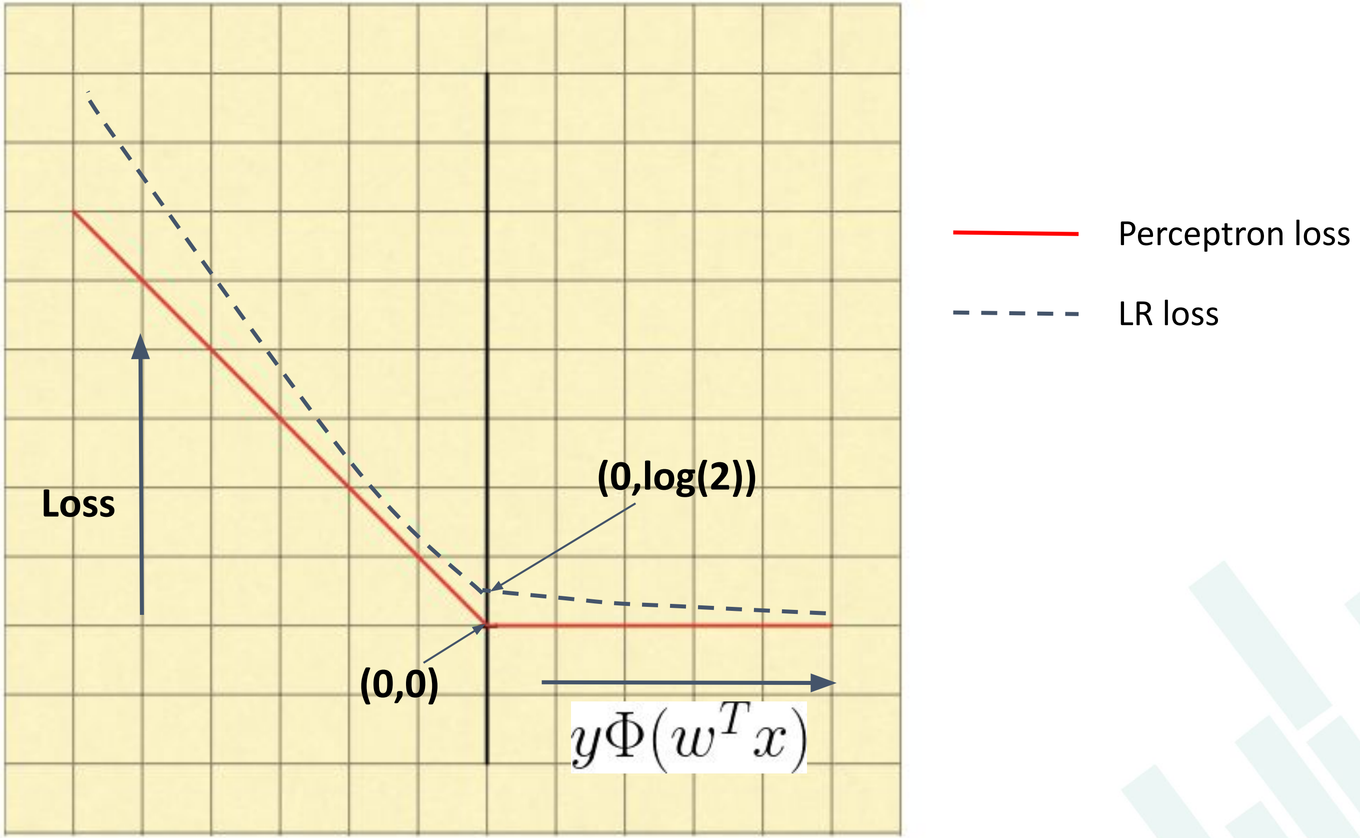
Loss function for perceptron

Φ = sigmoid function

$$\mathcal{L}_{\text{lr}}(x, y) = \begin{cases} -y\Phi(w^T x) + log(1 + e^{y\Phi(w^T x)}) & \text{if } y = +1 \text{ (positive)} \\ log(1 + e^{-y\Phi(w^T x)}) & \text{if } y = -1 \text{ (negative)} \end{cases}$$

Loss function for logistic regression

Loss: Perceptron and Logistic Regression



References

1. https://www.acm.org/media-center/2019/march/turing-award-2018
2. Chapter 3, Neural Networks: A Comprehensive Foundation (2nd Edition) 2nd Edition by Simon Haykin