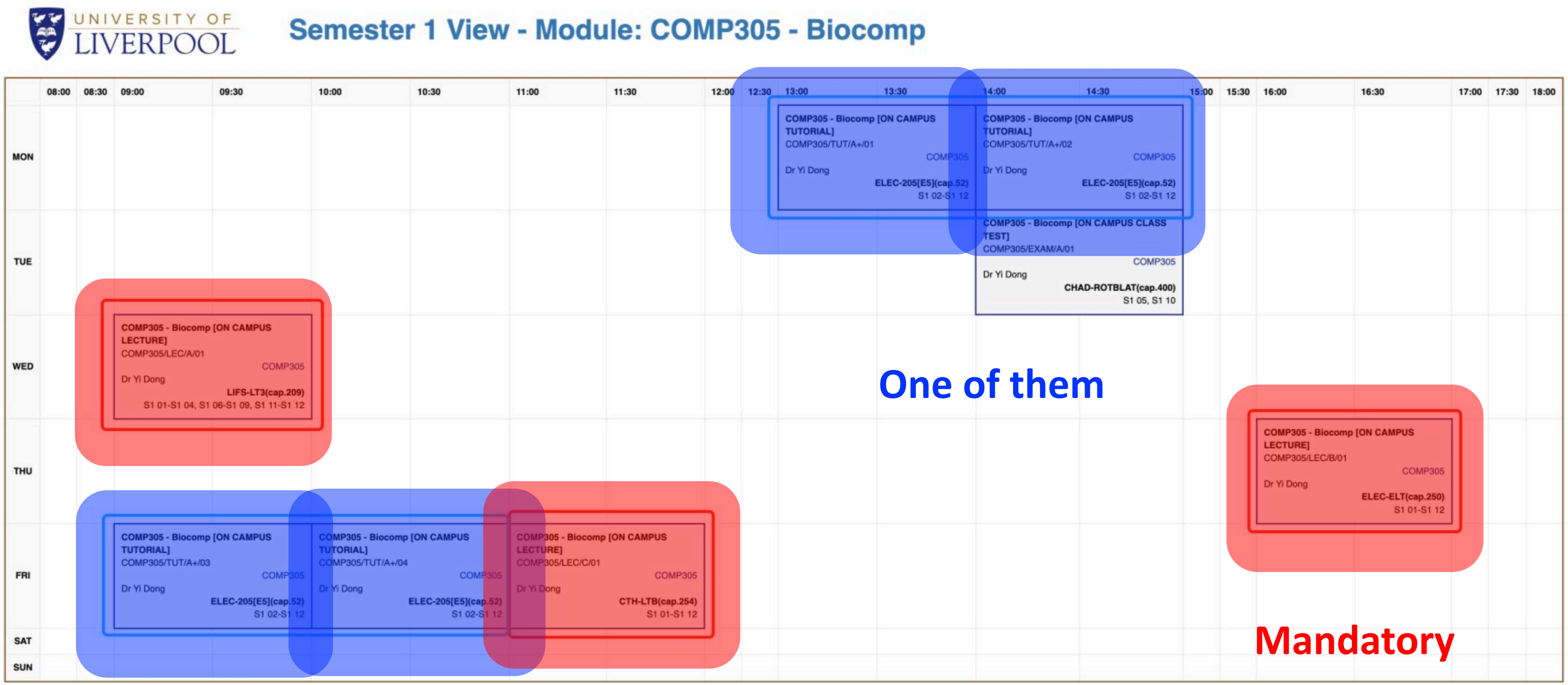
# Comp305

# Biocomputation

Lecturer: Yi Dong

# Comp305 Module Timetable




There will be **26-30** lectures, thee per week. The lecture slides will appear on Canvas. Please use Canvas to access the lecture information. There will be **9** tutorials, one per week.

# Lecture/Tutorial Rules

Questions are welcome as soon as they arise, because

1. Questions give feedback to the lecturer;

2. Questions help your understanding;

3. Your questions help your classmates, who might experience difficulties with formulating the same problems/doubts in the form of a question.

# Comp305 Part I.

# Artificial Neural Networks

# Topic 4.

# Perceptron

# Perceptron Learning Algorithm

**Algorithm 1: Perceptron Learning Algorithm**

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Learning rate $C$.

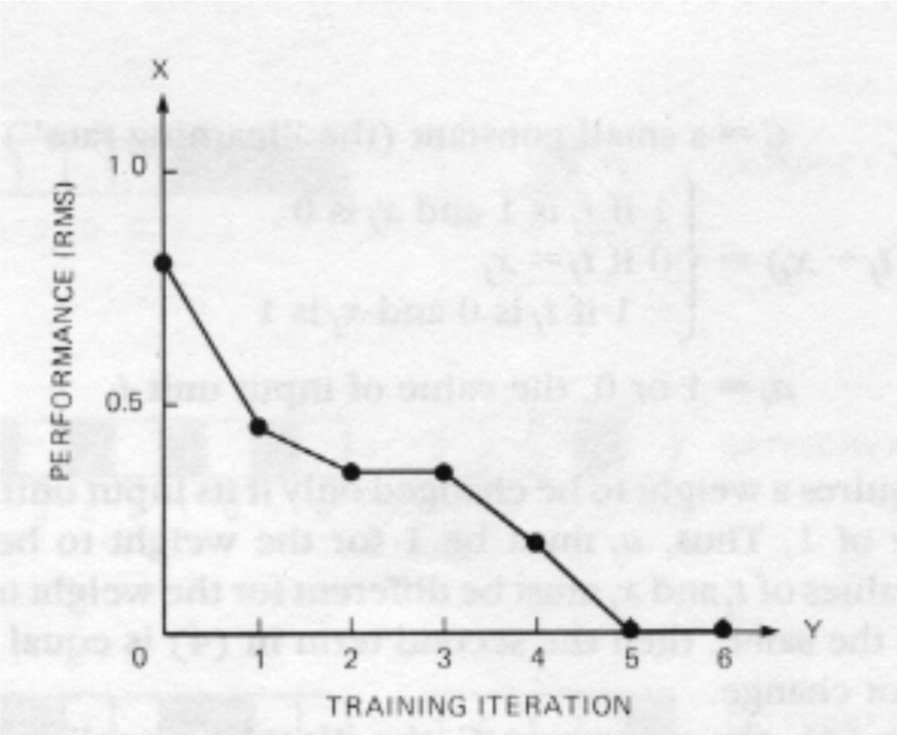
**Result:** Weight matrix $w = [w_1, \cdots, w_m]$

```
1 Initialize weights w randomly;
2 while !convergence (RMS ≤ δ) do
3     Pick random a' ∈ D;
4     a ⟵ [1, a'];
5     for j = 1, ··· , m do
          /* We represent the learning rule in the vector form   */
6         w_j = w_j + C(t_j − X_j)a;
7 return w;
```

Then the convergence checking is only done after one epoch.

A common way is to enumerate all the patterns in *D* sequentially. An epoch means training the neural network with all the training data for one cycle.

# Network Performance




**Q**: Does the learning rule always converge?

**Learning curve**: dependency of the RMS error on the number of iterations.

- ***Initially***, the adaptable ***weights are all set to small random values***, and the network does not perform very well;
- Performance improves **during training**;
- Finally, the error gets close to zero, training stops. We say the network has **converged**.

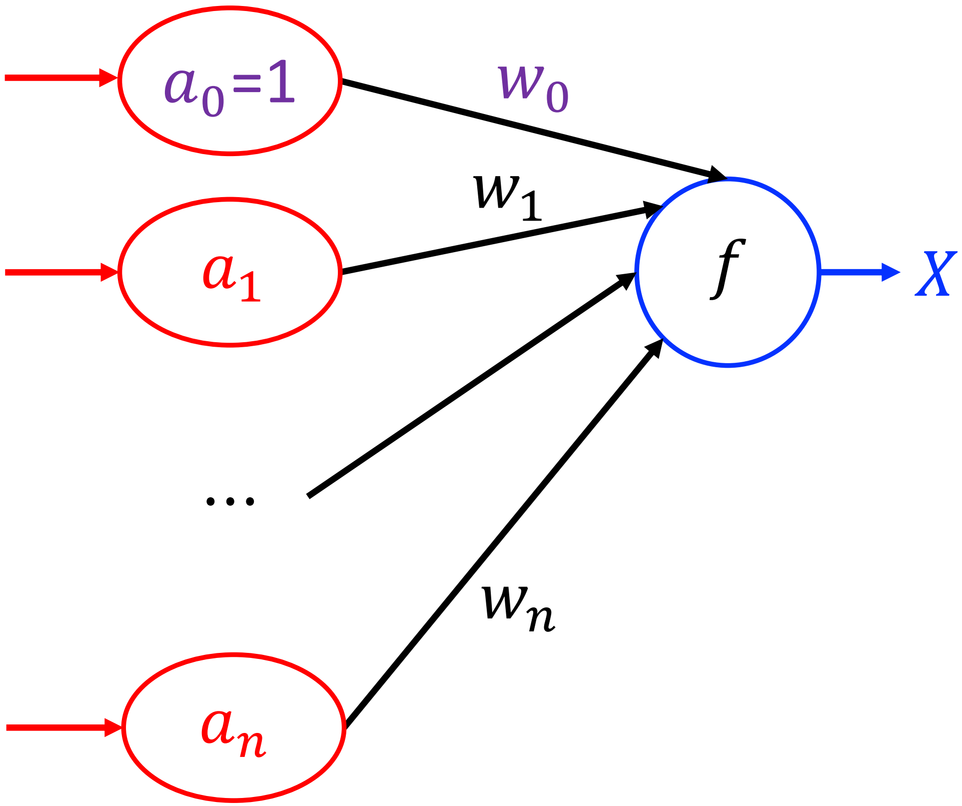
# Topic of Today's Lecture

Convergence of Perceptron Learning Algorithm

# Conclusion

The perceptron learning algorithm
can converge for the data set that is
linearly separable.

# Convergence of Perceptron Learning Algorithm




Without loss of generality, we consider a simplified case.

- There is only one output in the network,
- The learning rate $C$ is set as 1.

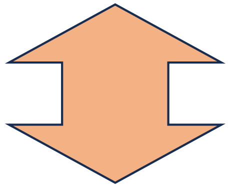
# Convergence of Perceptron Learning Algorithm

- Recall the following informal definition we mentioned for MP neuron.
- **Linear separability** (for Boolean functions): There exists a line (plane) such that all inputs which produce a 1 for the function lie on one side of the line (plane) and all inputs which produce a 0 lie on other side of the line (plane).

- Definition: Two sets $P$ and $N$ of points in an $n$-dimensional space are called (absolutely) **linearly separable** if there exists a real vector $w = (w_1, \cdots, w_n)$ such that every point $a' = (a_1, \cdots, a_n) \in P$ satisfies $w^T a' > 0$ and every point $a' = (a_1, \cdots, a_n) \in N$ satisfies $w^T a' < 0$.

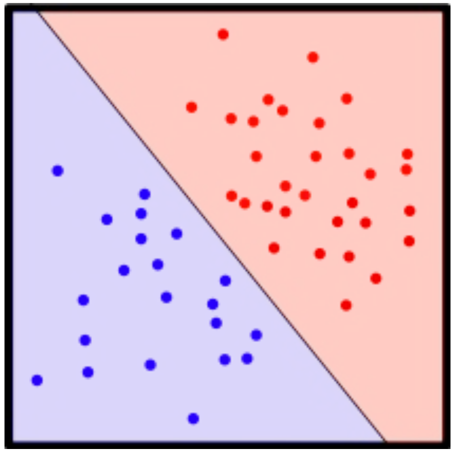
# Convergence of Perceptron Learning Algorithm

- Recall the following informal definition we mentioned for MP neuron.
- **Linear separability** (for Boolean functions): There exists a line (plane) such that all inputs which produce a 1 for the function lie on one side of the line (plane) and all inputs which produce a 0 lie on other side of the line (plane).



- Definition: Two sets $P$ and $N$ of points in an $n$-dimensional space are called (absolutely) **linearly separable** if there exists a real vector $w = (w_1, \cdots, w_n)$ such that every point $a' = (a_1, \cdots, a_n) \in P$ satisfies $w^T a' > 0$ and every point $a' = (a_1, \cdots, a_n) \in N$ satisfies $w^T a' < 0$.

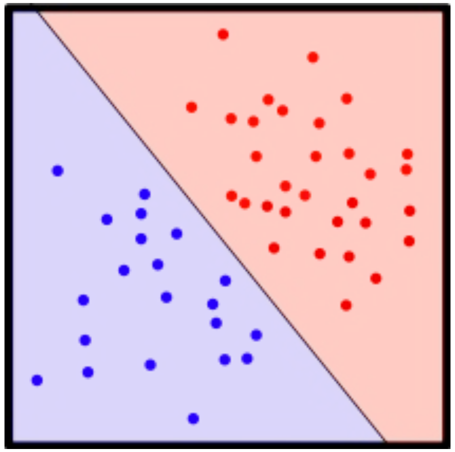
# Convergence of Perceptron Learning Algorithm



Source: wiki

- Definition: Two sets $P$ and $N$ of points in an $n$-dimensional space are called (absolutely) **linearly separable** if there exists a real vector $w = (w_1, \cdots, w_n)$ such that every point $a' = (a_1, \cdots, a_n) \in P$ satisfies $w^T a' > 0$ and every point $a' = (a_1, \cdots, a_n) \in N$ satisfies $w^T a' < 0$.

# Convergence of Perceptron Learning Algorithm

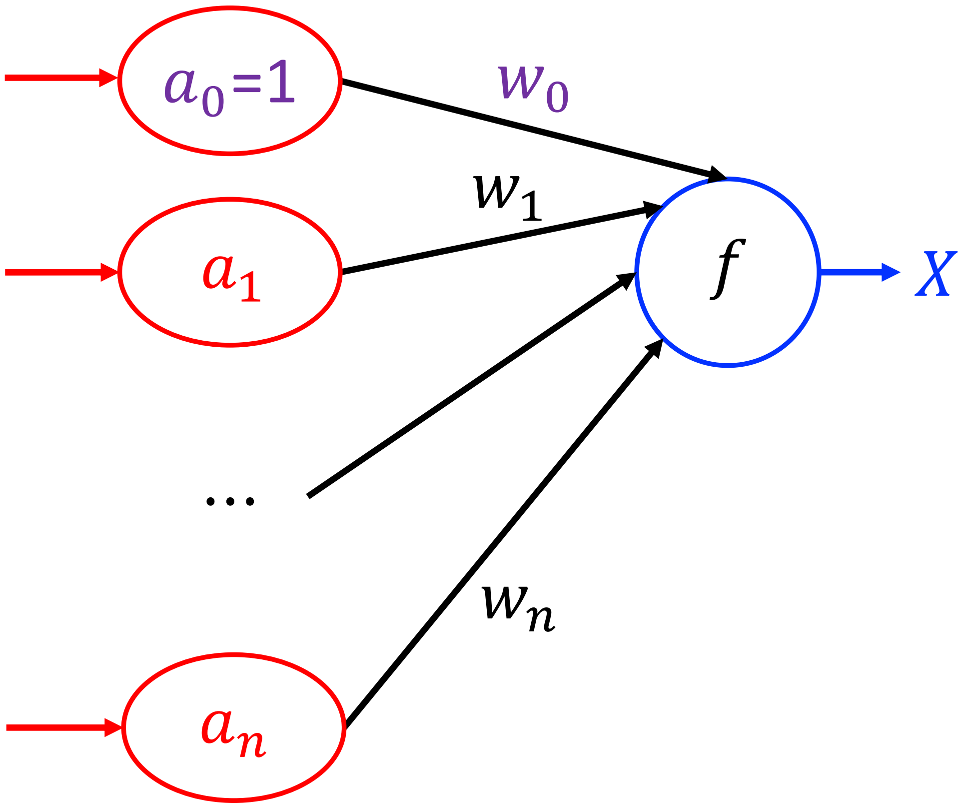


Source: wiki

- Definition: Two sets $P$ and $N$ of points in an $n$-dimensional space are called (absolutely) **linearly separable** if there exists a real vector $w = (w_1, \cdots, w_n)$ such that every point $a' = (a_1, \cdots, a_n) \in P$ satisfies $w^T a' > 0$ and every point $a' = (a_1, \cdots, a_n) \in N$ satisfies $w^T a' < 0$.

If in the data set $D$, 1-label subset and 0-label subset are linearly separable, we say $D$ is (absolutely) **linearly separable**.

# Convergence of Perceptron Learning Algorithm




Without loss of generality, we consider a simplified case.

- There is only one output in the network,
- The learning rate $C$ is set as 1.

We assume the data set $D$ is (absolutely) **linearly separable**.

# Perceptron Learning Algorithm (One output)

We assume the data set $D$ is (absolutely) **linearly separable**.

The set $D' = \{[1, a'] | a' \in D\}$ is also (absolutely) **linearly separable**.
(**Why?**)

# Perceptron Learning Algorithm (One output)

We assume the data set $D$ is (absolutely) **linearly separable**.

The set $D' = \{[1, a'] | a' \in D\}$ is also (absolutely) **linearly separable**.
(**Why?**)

Tip: Let $w_0 = 0$

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix~~ $\cancel{w = [w_0, \cdots, w_m]}$

$P \longleftarrow$ Inputs with label 1;
$N \longleftarrow$ Inputs with label 0;
Initialize weights $w$ randomly;
**while** *!convergence (RMS $\leq \delta$)* **do**
    Pick random $a' \in D$;
    $a \longleftarrow [1, a']$;
    **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
        /* $t = 1$, learning rate is 1 */
        $w = w + a$;
    **end**
    **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
        /* $t = 0$, learning rate is 1 */
        $w = w - a$;
    **end**
**end**
**return** $w$;

Rewriting the general perceptron learning algorithm.

- Since we only consider one output, the result becomes a weight vector

$$w = (w_0, w_1, \cdots, w_n)$$

rather than the weight matrix. Here $w_i$ represents for the weight of the connection between the $i$-th input and the output.

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

```
P ⟵ Inputs with label 1;
N ⟵ Inputs with label 0;
Initialize weights w randomly;
while !convergence (RMS ≤ δ) do
    Pick random a' ∈ D;
    a ⟵ [1, a'];
    if a ∈ P and X = 0 (w^T a < 0) then
        /* t = 1, learning rate is 1 */
        w = w + a;
    end
    if a ∈ N and X = 1 (w^T a ≥ 0) then
        /* t = 0, learning rate is 1 */
        w = w − a;
    end
end
return w;
```

Rewriting the general perceptron learning algorithm.

- The data set $D' = \{[1, a'] | a' \in D\}$ can be divided into two separated set $P$ and $N$, by the definition of absolutely linear separability.

That is, $D' = P \cup N, P \cap N = \emptyset$, and there exists a real vector $w^* = (w_0, w_1, \cdots, w_n)$ exist such that every point $a = (a_0, a_1, \cdots, a_n) \in P$ satisfies ${w^*}^T a > 0$ and every point $a = (a_0, a_1, \cdots, a_n) \in N$ satisfies ${w^*}^T a \ < 0$.

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

1 $P \longleftarrow$ Inputs with label 1;
2 $N \longleftarrow$ Inputs with label 0;
3 Initialize weights $w$ randomly;
4 **while** *!convergence (RMS $\leq \delta$)* **do**
5     Pick random $a' \in D$;
6     $a \longleftarrow [1, a']$;
7     **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
        `/* t = 1, learning rate is 1 */`
8         $w = w + a$;
9     **end**
10     **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
        `/* t = 0, learning rate is 1 */`
11         $w = w - a$;
12     **end**
13 **end**
14 **return** $w$;

Rewriting the general perceptron learning algorithm.

- The data set $D' = \{[1, a'] | a' \in D\}$ can be divided into two separated set $P$ and $N$, by the definition of absolutely linear separability. (Line 1, Line 2)

That is, $D' = P \cup N, P \cap N = \emptyset$, and there exists a real vector $w^* = (w_0, w_1, \cdots, w_n)$ exist such that every point $a = (a_0, a_1, \cdots, a_n) \in P$ satisfies ${w^*}^T a > 0$ and every point $a = (a_0, a_1, \cdots, a_n) \in N$ satisfies ${w^*}^T a < 0$.

The convergence of the learning rule means we successfully find a feasible $w^*$!

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix~~ $\cancel{w = [w_0, \cdots, w_m]}$

1 $P \longleftarrow$ Inputs with label 1;
2 $N \longleftarrow$ Inputs with label 0;
3 Initialize weights $w$ randomly;
4 **while** *!convergence (RMS $\leq \delta$)* **do**
5     Pick random $a' \in D$;
6     $a \longleftarrow [1, a']$;
7     **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
        /* $t = 1$, learning rate is 1 */
8         $w = w + a$;
9     **end**
10     **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
        /* $t = 0$, learning rate is 1 */
11         $w = w - a$;
12     **end**
13 **end**
14 **return** $w$;

Rewriting the general perceptron learning algorithm.

- The data set $D' = \{[1, a'] | a' \in D\}$ can be divided into two separated set $P$ and $N$, by the definition of absolutely linear separability.
- We consider the different cases in terms of the input pattern and the corresponding output value. (Line 7 - 12)

$$\Delta w_{ji}^k = C e_j^k a_i^k$$

$$e_j^k = t_j^k - X_j^k = \begin{cases} \boxed{1, \quad t_j^k = 1, X_j^k = 0} \\ 0, \quad t_j^k = X_j^k \\ \boxed{-1, \quad t_j^k = 0, X_j^k = 1} \end{cases}$$

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

1 $P \longleftarrow$ Inputs with label 1;
2 $N \longleftarrow$ Inputs with label 0;
3 Initialize weights $w$ randomly;
4 **while** *!convergence (RMS $\leq \delta$)* **do**
5     Pick random $a' \in D$;
6     $a \longleftarrow [1, a']$;
7     **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
        /* $t=1$, learning rate is 1 */
8         $w = w + a$;
9     **end**
10     **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
        /* $t=0$, learning rate is 1 */
11         $w = w - a$;
12     **end**
13 **end**
14 **return** $w$;

Rewriting the general perceptron learning algorithm.

- The data set $D' = \{[1, a'] | a' \in D\}$ can be divided into two separated set $P$ and $N$, by the definition of absolutely linear separability. (Line 1, Line 2)
- We consider the different cases in terms of the input pattern and the corresponding output value . (Line 7 - 12)

$$\Delta w_{ji}^k = C e_j^k a_i^k$$

$$e_j^k = t_j^k - X_j^k = \begin{cases} 1, & t_j^k = 1, X_j^k = 0 \\ 0, & t_j^k = X_j^k \\ -1, & t_j^k = 0, X_j^k = 1 \end{cases}$$

# Perceptron Learning Algorithm (One output)

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P \longleftarrow$ Inputs with label 1;

$N \longleftarrow$ Inputs with label 0;

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

  **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**

    /* $t = 0$, learning rate is 1 */

    $w = w - a$;

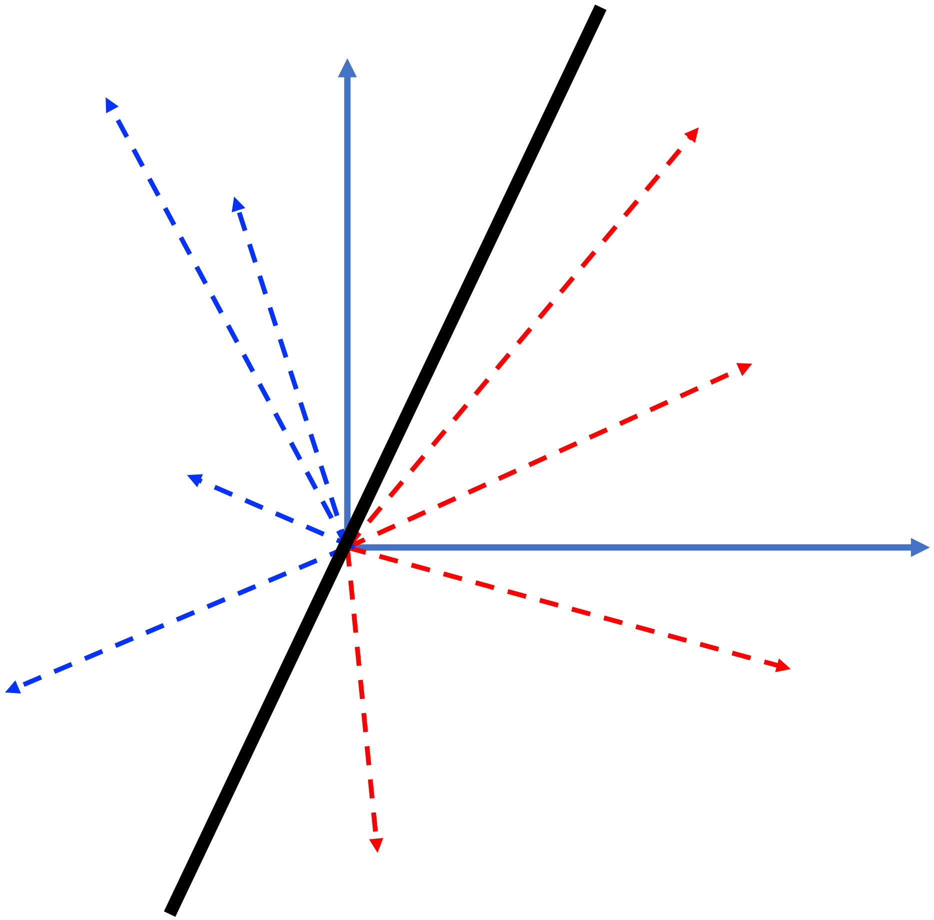
  **end**

**end**

**return** $w$;

**We first explore the intuition of the learning algorithm.**

# Geometric Interpretation

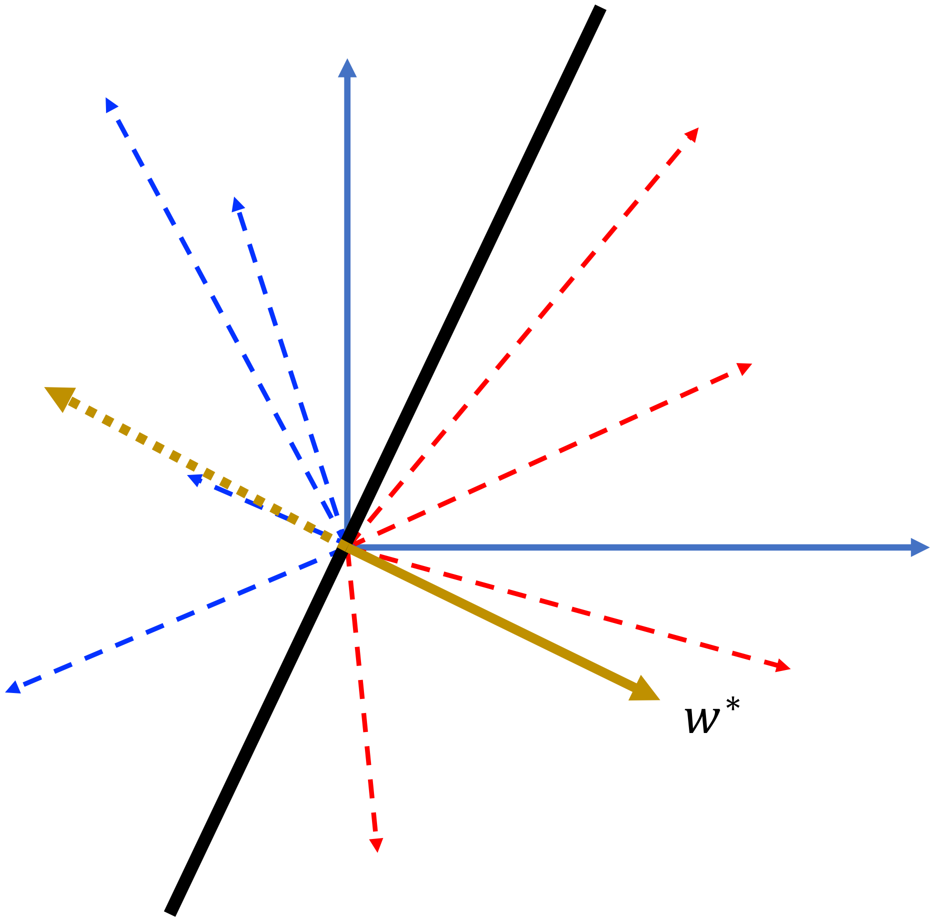


We plot all the input patterns $a$.

- Red arrows denote the points in $P$,
- Blue arrows denote the points in $N$,
- **Thick black line** denotes the barrier of ${w^*}^T a = 0$

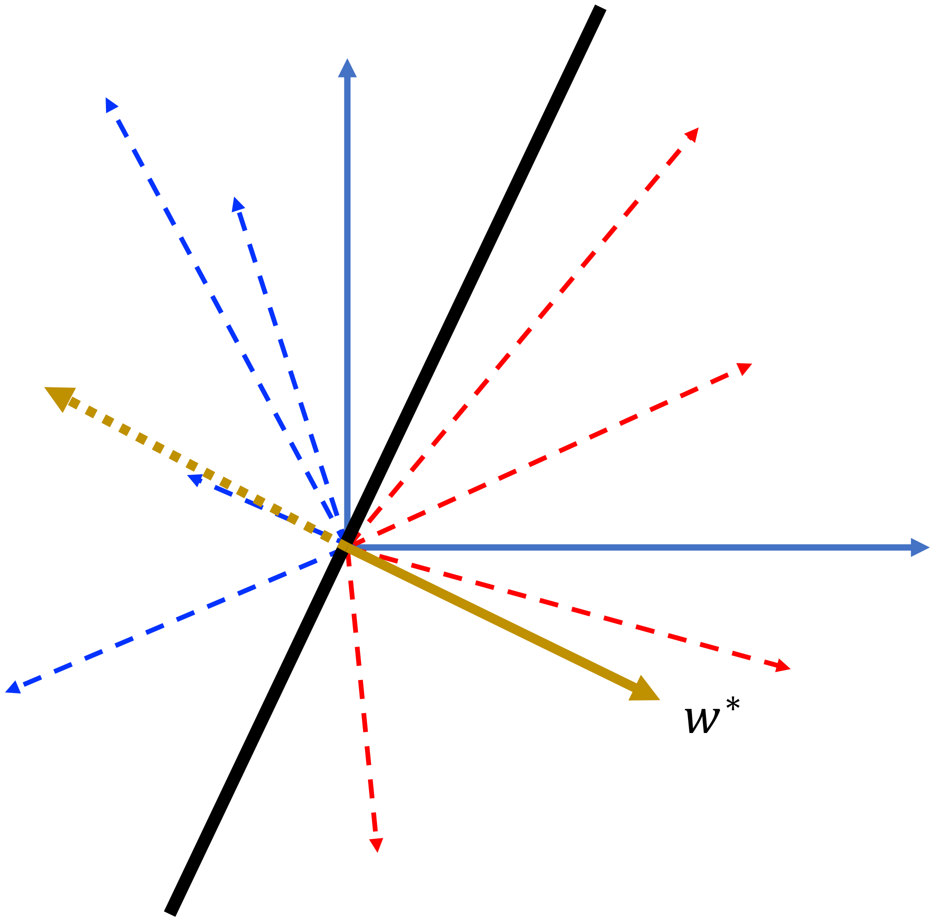
We can do this, due to the definition of absolutely linear separability.

# Geometric Interpretation




Meanwhile, ${w^*}^T a = 0$ means that the vector $w^*$ and the barrier are **orthogonal**.

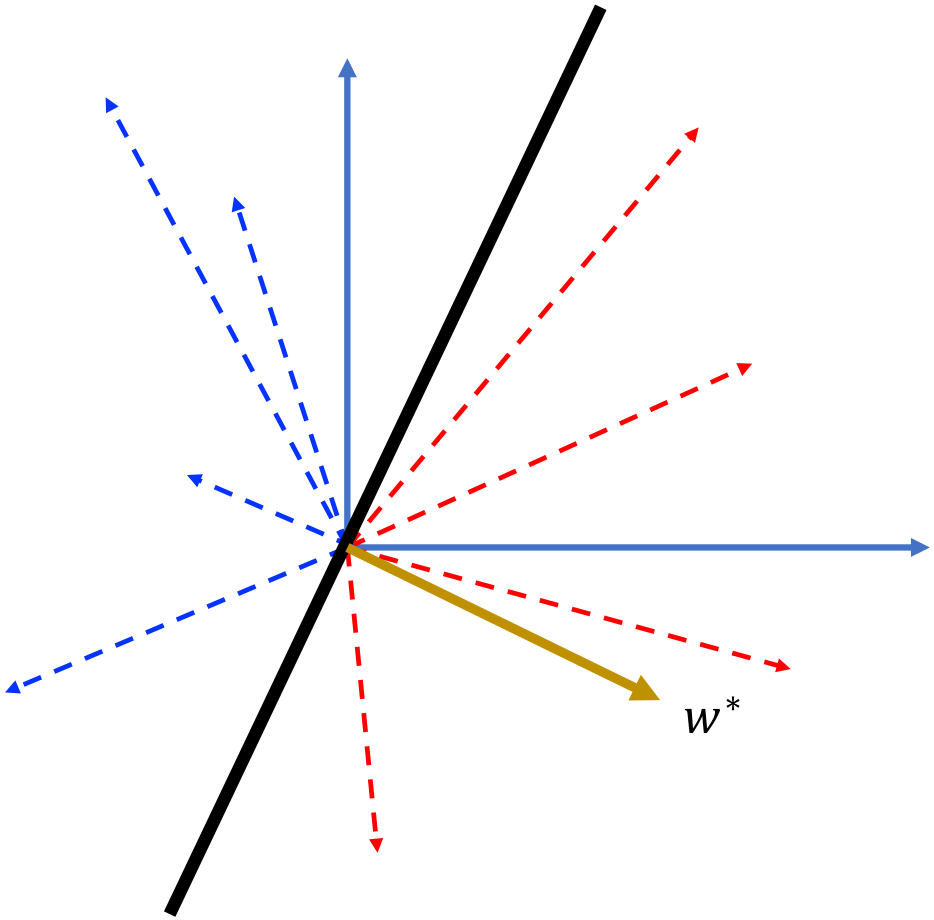
# Geometric Interpretation




Meanwhile, ${w^*}^T a = 0$ means that the vector $w^*$ and the barrier are **orthogonal**.

**Questions**: There are two orthogonal vectors. Why don't we choose the dashed one?

# Geometric Interpretation



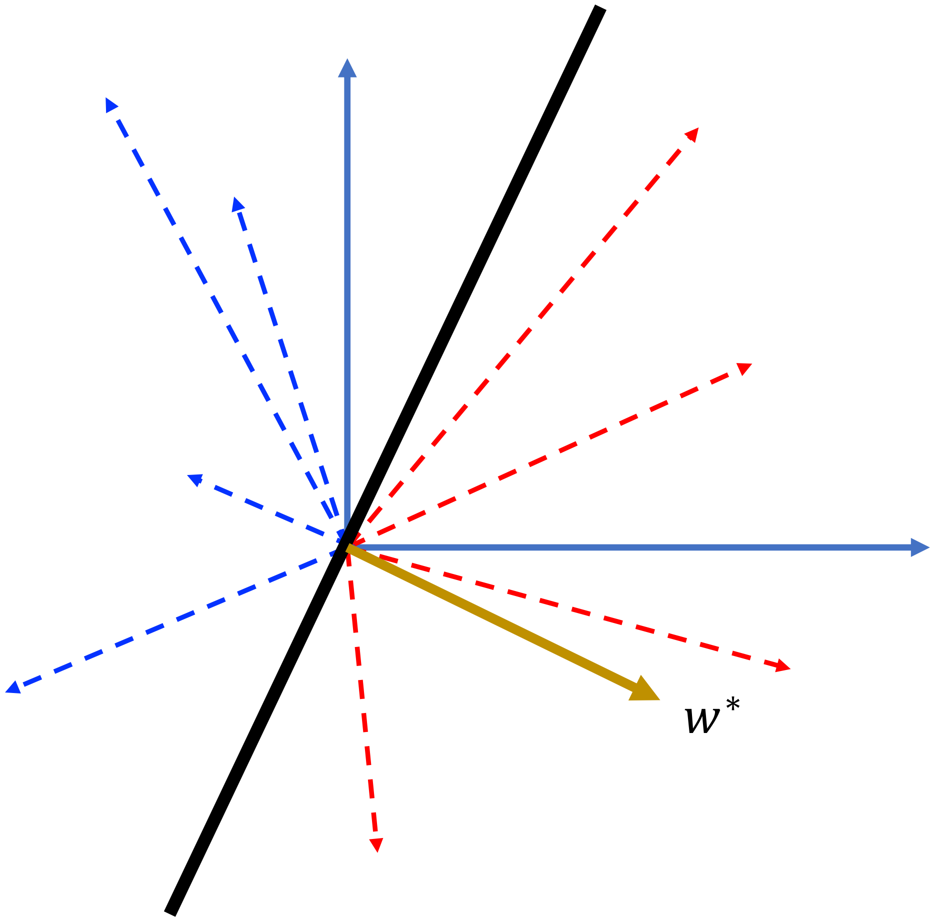

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\|\|\boldsymbol{b}\| \cos\langle \boldsymbol{a}, \boldsymbol{b} \rangle$$

Meanwhile, ${w^*}^T a = 0$ means that the vector $w^*$ and the barrier are **orthogonal**.

For all $a \in P$, ${w^*}^T a > 0$ means that the angle between the vector $w^*$ and every input pattern in $P$ is **less than 90°.**

# Geometric Interpretation

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \|\boldsymbol{b}\| \cos\langle \boldsymbol{a}, \boldsymbol{b} \rangle$$



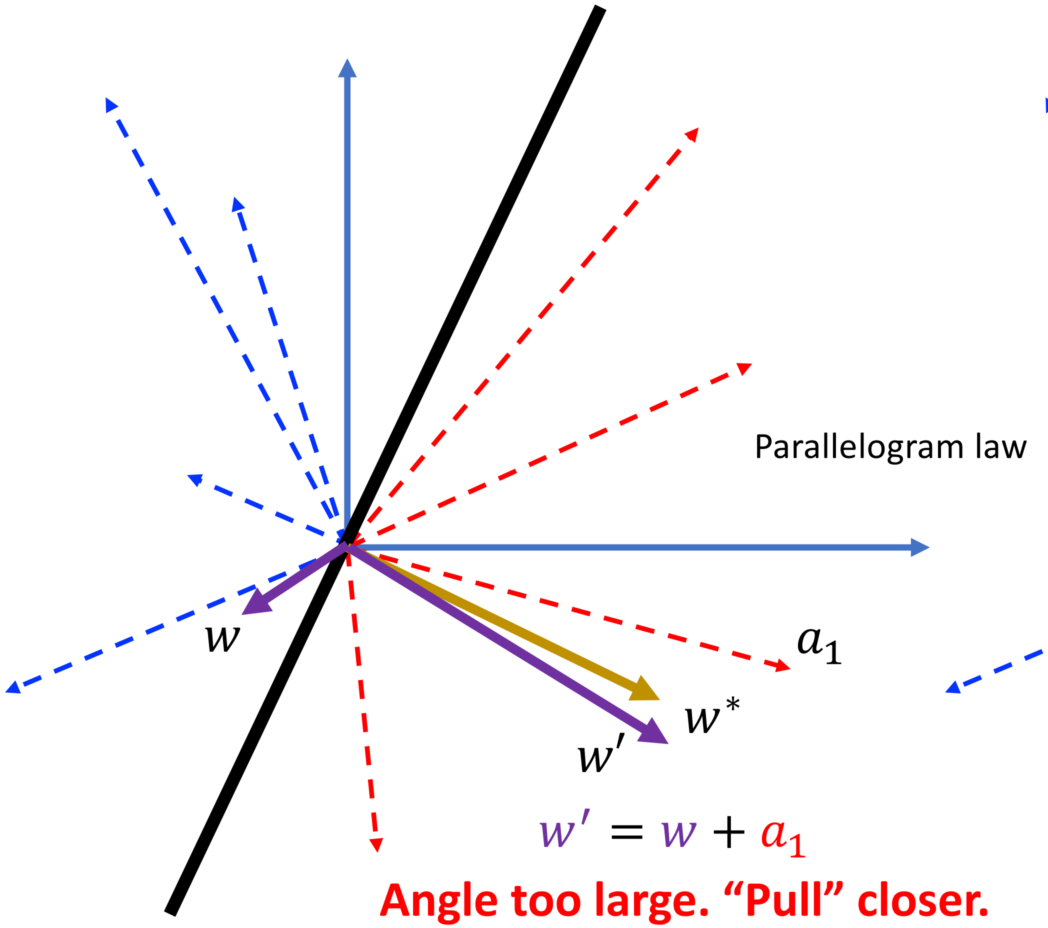

Meanwhile, ${w^*}^T a = 0$ means that the vector $w^*$ and the barrier are **orthogonal**.

For all $a \in P$, ${w^*}^T a > 0$ means that the angle between the vector $w^*$ and every input pattern in $P$ is **less than 90°.**

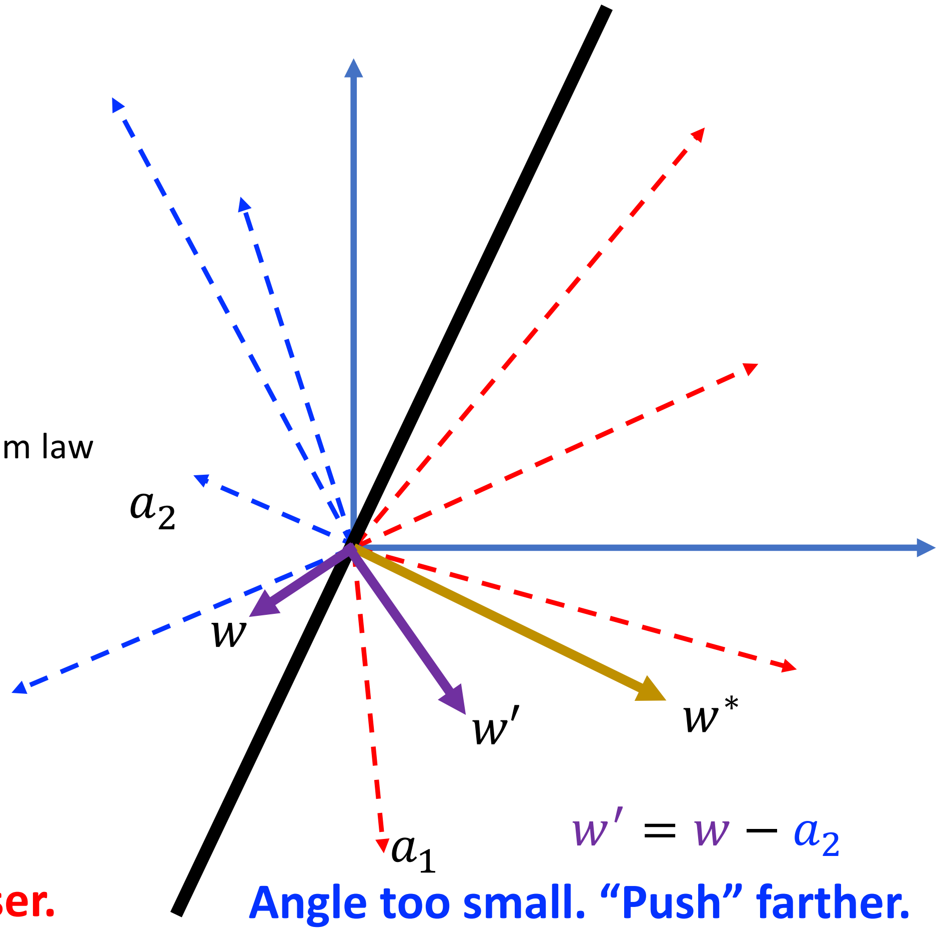
For all $a \in N$, ${w^*}^T a < 0$ means that the angle between the vector $w^*$ and every input pattern in $N$ is **greater than 90°.**

# Geometric Interpretation

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \|\boldsymbol{b}\| \cos\langle \boldsymbol{a}, \boldsymbol{b} \rangle$$




**Angle too large. "Pull" closer.**




**Angle too small. "Push" farther.**

# Geometric Interpretation

---

**Algorithm 2: Perceptron Learning** (One output)

---

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix~~ $\cancel{w = [w_0, \cdots, w_m]}$

```
P ⟵ Inputs with label 1;
N ⟵ Inputs with label 0;
Initialize weights w randomly;
while !convergence (RMS ≤ δ) do
    Pick random a′ ∈ D;
    a ⟵ [1, a′];
    if a ∈ P and X = 0 (w^T a < 0) then
        /* t = 1, learning rate is 1    */
        w = w + a;
    end
    if a ∈ N and X = 1 (w^T a ≥ 0) then
        /* t = 0, learning rate is 1    */
        w = w − a;
    end
end
return w;
```

---

This algorithm means that

- in each iteration, we "pull" the weight vector $w$ closer to $P$, "push" the weight vector $w$ farther to $N$, if misclassified.
- until the angle between $w$ and each pattern in $P$ is **less than $90°$**, and the angle between $w$ and each pattern in $N$ is **greater than $90°$**.
- In both cases, $w$ gets closer to $w^*$.

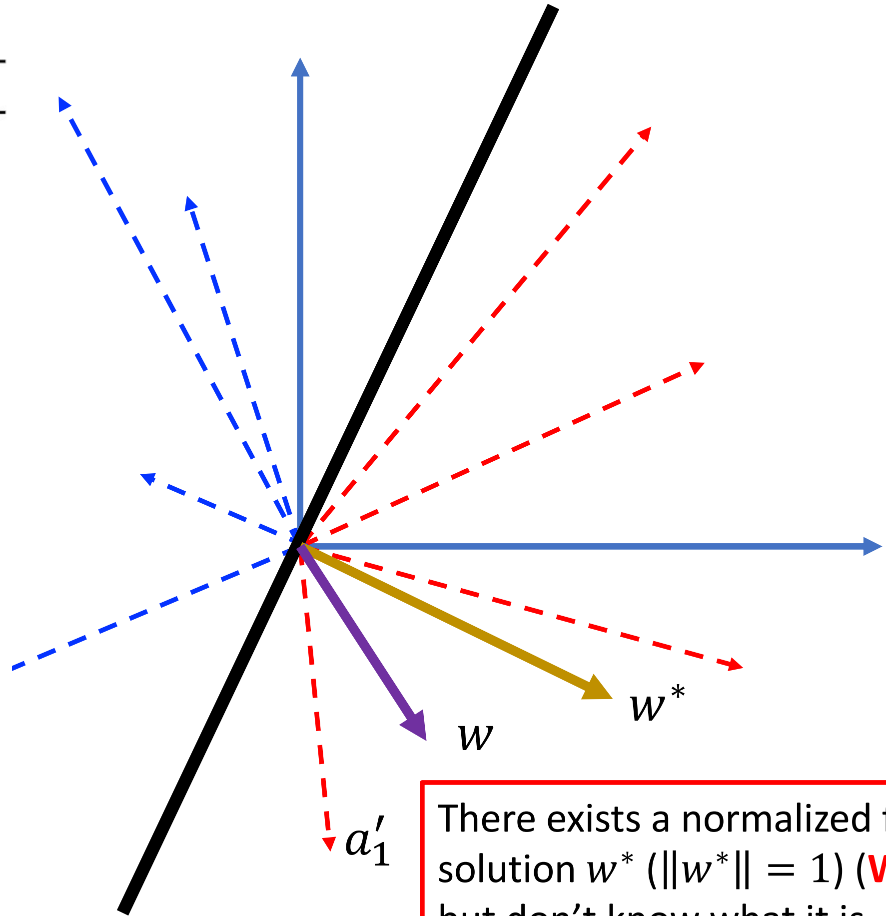
# Geometric Interpretation

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P \longleftarrow$ Inputs with label 1;
$N \longleftarrow$ Inputs with label 0;
Initialize weights $w$ randomly;
**while** *!convergence (RMS $\leq \delta$)* **do**
  Pick random $a' \in D$;
  $a \longleftarrow [1, a']$;
  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
    /* $t = 1$, learning rate is 1 */
    $w = w + a$;
  **end**
  **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
    /* $t = 0$, learning rate is 1 */
    $w = w - a$;
  **end**
**end**
**return** $w$;




There exists a normalized feasible solution $w^*$ ($\|w^*\| = 1$) (**Why?**), but don't know what it is.

# Geometric Interpretation

---

**Algorithm 2: Perceptron Learning** (One output)

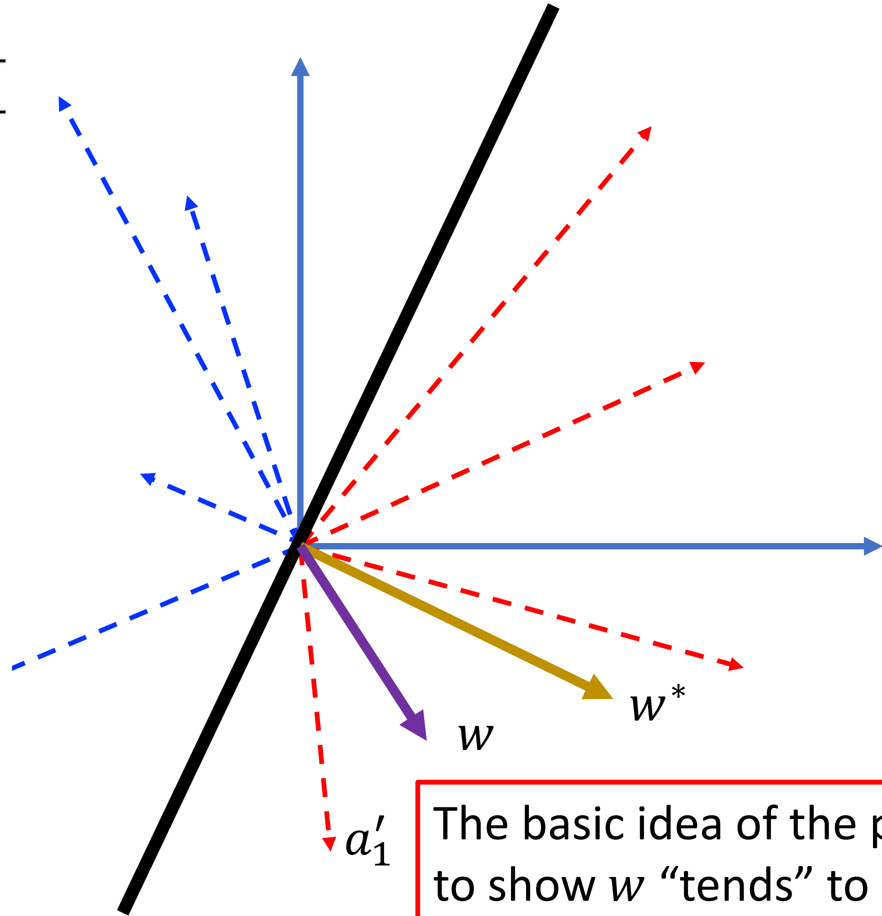
---

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P \longleftarrow$ Inputs with label 1;
$N \longleftarrow$ Inputs with label 0;
Initialize weights $w$ randomly;
**while** *!convergence (RMS $\leq \delta$)* **do**
  Pick random $a' \in D$;
  $a \longleftarrow [1, a']$;
  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
    /* $t = 1$, learning rate is 1 */
    $w = w + a$;
  **end**
  **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
    /* $t = 0$, learning rate is 1 */
    $w = w - a$;
  **end**
**end**
**return** $w$;

---




The basic idea of the proof is to show $w$ "tends" to get closer to $w^*$ during training.

# Convergence Analysis

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P \longleftarrow$ Inputs with label 1;
$N \longleftarrow$ Inputs with label 0;
Initialize weights $w$ randomly;
**while** *!convergence (RMS $\leq \delta$)* **do**
    Pick random $a' \in D$;
    $a \longleftarrow [1, a']$;
    **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
        /* $t = 1$, learning rate is 1 */
        $w = w + a$;
    **end**
    **if** $a \in N$ *and* $X = 1$ $(w^T a \geq 0)$ **then**
        /* $t = 0$, learning rate is 1 */
        $w = w - a$;
    **end**
**end**
**return** $w$;

Now we start to prove the convergence.

Without loss of generality, we assume that $D' = P$. (**why?**)

$\boxed{\text{Let } \boldsymbol{P}' = -\boldsymbol{N}.\ P = P \cup \boldsymbol{P}'.}$

# Convergence Analysis

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Now we start to prove the convergence.

Without loss of generality, we assume that $D' = P$. (**why?**)

$\boxed{\text{Let } \boldsymbol{P'} = -\boldsymbol{N}.\ P = P \cup \boldsymbol{P'}.}$

# Convergence Analysis

---

**Algorithm 2: Perceptron Learning** (One output)

---

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

```
P = D
Initialize weights w randomly;
while !convergence (RMS ≤ δ) do
    Pick random a′ ∈ D;
    a ⟵ [1, a′];
    if a ∈ P and X = 0 (w^T a < 0) then
        /* t = 1, learning rate is 1 */
        w = w + a;
    end
end
return w;
```

---

Now we start to prove the convergence.

Without loss of generality, we assume that $D' = P$. (**why?**)

$\boxed{\text{Let } \boldsymbol{P'} = -\boldsymbol{N}.\ P = P \cup \boldsymbol{P'}.}$

Without loss of generality, we assume that for each $a'$, $\|a'\| = 1$ . (**why?**)

$$\boxed{\|a'\| = 1 \Leftrightarrow \|a\| = \sqrt{2}}$$

# Convergence Analysis

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$$P = D$$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

- As we mentioned, we focus on how close the current $w^{k+1}$ and the solution $w^*$, which can be evaluated by the angle $\theta^{k+1}$ between them.

$$\cos\theta^{k+1} = \frac{w^{k+1} \cdot w^*}{\|w^{k+1}\| \cdot \|w^*\|} = \frac{w^{k+1} \cdot w^*}{\|w^{k+1}\|}$$

**Getting closer** means ***cos θ* becomes bigger**.

So, what we are going to do is to

- Check the denominator $\|w^{k+1}\|$,
- Check the numerator $w^{k+1} \cdot w^*$.

# Convergence Analysis

$$\boxed{\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$\boxed{P = D}$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ ($w^T a < 0$) **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

Check the denominator $\|w^{k+1}\|$.

- If at $k$-th iteration, the input pattern $a^k$ is misclassified, that is, $w^k \cdot a^k < 0$. By the algorithm we know $w^{k+1} = w^k + a^k$.

$$\|w^{k+1}\|$$
$$= \sqrt{(w^k + a^k)^2}$$
$$= \sqrt{\|w^k\|^2 + 2w^k \cdot a^k + \|w^k\|^2}$$
$$< \sqrt{\|w^k\|^2 + \|a^k\|^2}$$
$$= \sqrt{\|w^k\|^2 + 2}$$

Dot product

$\boxed{w^k \cdot a^k < 0}$

Assumption

- If the input pattern $a_k$ is classified correctly.

$$\|w^{k+1}\| = \|w^k\|$$

# Convergence Analysis

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;
**while** *!convergence (RMS $\leq \delta$)* **do**
  Pick random $a' \in D$;
  $a \longleftarrow [1, a']$;
  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**
    /* $t = 1$, learning rate is 1 */
    $w = w + a$;
  **end**
**end**
**return** $w$;

Proof.

Check the denominator $\|w^{k+1}\|$.

- At $k$-th iteration,

$$\underbrace{\|w^{k+1}\| < \sqrt{\|w^k\|^2 + 2}}_{\text{Misclassified}} \ \textbf{or} \ \underbrace{\|w^{k+1}\| = \|w^k\|}_{\text{Correctly classified}}$$

Inductively, we know

$$\|w^{k+1}\| < \sqrt{\|w^1\|^2 + 2k'}$$

where $k'$ is the number of misclassified iterations among $k$ iterations.

We got an **upper bound** of the denominator.

# Convergence Analysis

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

Check the numerator $w^{k+1}\cdot w^*$.

- If at $k$-th iteration, the input pattern $a_k$ is misclassified, that is, $w^k\cdot a^k < 0$. By the algorithm we know $w^{k+1} = w^k + a^k$.

$$\begin{aligned} & w^{k+1}\cdot w^* \\ &= \left(w^k + a^k\right)\cdot w^* \\ &= w^k\cdot w^* + a_k\cdot w^* \\ &\geq w^k\cdot w^* + \alpha \end{aligned}$$

$\forall a_k, w^*\cdot a^k > 0$,
Let $\alpha = \min\{(w^*)^T a^k\} > 0$

- If the input pattern $a_k$ is classified correctly.

$$w^{k+1}\cdot w^* = w^k\cdot w^*$$

# Convergence Analysis

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t=1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

Check the numerator $w^{k+1} \cdot w^*$.

- At $k$-th iteration,

$$w^{k+1}\cdot w^* \geq w^k \cdot w^* + \alpha \text{ or } w^{k+1}\cdot w^* = w^k \cdot w^*$$

Misclassified Correctly classified

Inductively, we know

$$w^{k+1}\cdot w^* \geq w^1 \cdot w^* + k'\alpha$$

where $k'$ is the number of misclassified iterations among $k$ iterations.

We got a **lower bound** of the numerator.

# Convergence Analysis

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

---

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

```
P = D

Initialize weights w randomly;
while !convergence (RMS ≤ δ) do
    Pick random a′ ∈ D;
    a ⟵ [1, a′];
    if a ∈ P and X = 0 (w^T a < 0) then
        /* t = 1, learning rate is 1     */
        w = w + a;
    end
end
return w;
```

---

Proof.

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|\cdot\|w^*\|} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

- Check the denominator $\|w^{k+1}\|$,

$$\|w^{k+1}\| \le \sqrt{\|w^1\|^2 + 2k'}$$

- Check the numerator $w^{k+1}\cdot w^*$.

$$w^{k+1}\cdot w^* \ge w^1\cdot w^* + k'\alpha$$

Now we know

$$\cos\theta^{k+1} \ge \frac{w^1\cdot w^* + k'\alpha}{\sqrt{\|w^1\|^2 + 2k'}}$$

# Convergence Analysis

$$\boxed{\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|\cdot\|w^*\|} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

- Check the denominator $\|w^{k+1}\|$,

$$\|w^{k+1}\| \leq \sqrt{\|w^1\|^2 + 2k'}$$

- Check the numerator $w^{k+1}\cdot w^*$.

$$w^{k+1}\cdot w^* \geq w^1\cdot w^* + k'\alpha$$

Now we know

$$\cos\theta^{k+1} \geq \frac{w^1\cdot w^* + k'\alpha}{\sqrt{\|w^1\|^2 + 2k'}}$$

$\cos\theta^{k+1}$ grows proportional to $\sqrt{k'}$.

# Convergence Analysis

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

**Algorithm 2: Perceptron Learning** (One output)

**Data:** Labelled data set $D$: $r$ $n$-dimensional input points, each of which has $m$ labels. Small positive real $\delta$. Weight vector: $(w_0, w_1, \cdots, w_n)$

**Result:** ~~Weight matrix $w = [w_0, \cdots, w_m]$~~

$P = D$

Initialize weights $w$ randomly;

**while** *!convergence (RMS $\leq \delta$)* **do**

  Pick random $a' \in D$;

  $a \longleftarrow [1, a']$;

  **if** $a \in P$ *and* $X = 0$ $(w^T a < 0)$ **then**

    /* $t = 1$, learning rate is 1 */

    $w = w + a$;

  **end**

**end**

**return** $w$;

Proof.

$$\cos\theta^{k+1} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|\cdot\|w^*\|} = \frac{w^{k+1}\cdot w^*}{\|w^{k+1}\|}$$

Now we know

$$\cos\theta^{k+1} \geq \frac{w^1\cdot w^* + k'\alpha}{\sqrt{\|w^1\|^2 + 2k'}}$$

$\cos\theta^{k+1}$ grows proportional to $\sqrt{k'}$.

If the algorithm does not terminate, that is, there will be infinite misclassified inputs, $k'$ would go to infinity, $\cos\theta^{k+1}$ would also go to infinity, which is impossible.

$$\cos\theta^{k+1} \leq 1$$

# Beyond Linear Separability

- Now we know the perceptron learning algorithm can finally converge for the data set that is linearly separable.

- How about the one that is not linearly separable?