# The Perceptron

Kun He (何琨)

Data Mining and Machine Learning Lab
(John Hopcroft Lab)
Huazhong University of Science & Technology

*brooklet60@hust.edu.cn*

2022年05月

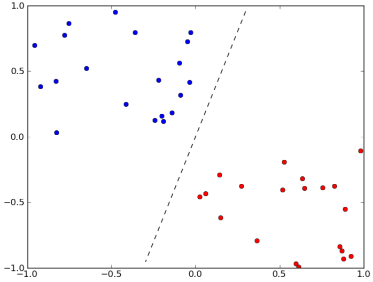
## Table of Contents

## Table of Contents

## Assumptions

### Assumptions

- Binary classification (i.e. $y_i \in \{-1, +1\}$)
- Data is linearly separable

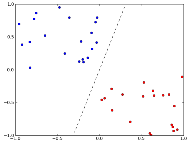


## Assumptions

**Basic idea:**

- In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers.
- A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.
- It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.

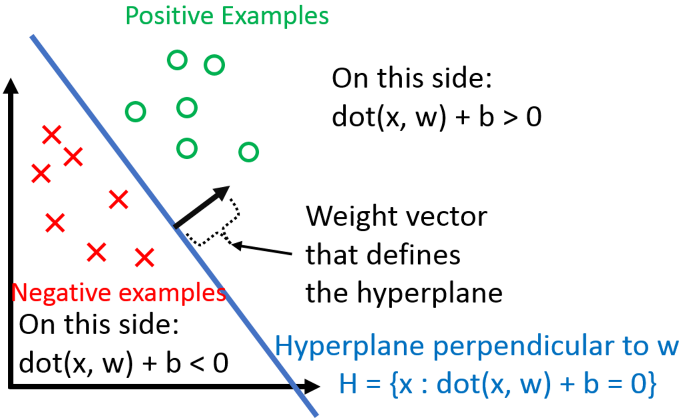
$$\mathcal{H} = \{h(x) = \mathbf{w}^\top \mathbf{x} + b = 0\}$$



## Table of Contents

## Parameter Selection

$$h(x_i) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$



## Parameter Selection

$b$ is the bias term (without the bias term, the hyperplane that $\mathbf{w}$ defines would always have to go through the origin). Dealing with $b$ can be a pain, so we 'absorb' it into the feature vector $\mathbf{w}$ by adding one additional constant dimension. Under this convention,

$$\mathbf{x}_i \quad \text{becomes} \quad \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$$

$$\mathbf{w} \quad \text{becomes} \quad \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

We can verify that

$$\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \mathbf{w}^\top \mathbf{x}_i + b$$

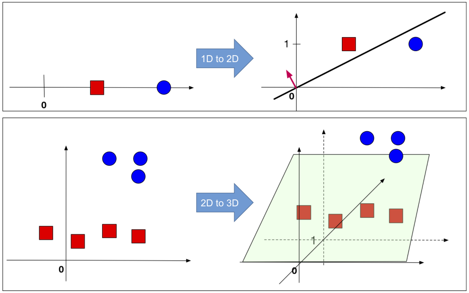
Hence,

$$\mathcal{H} = \{h(x) = \mathbf{w}^\top \mathbf{x} = 0\}$$

## Hyperplane

Then, we can simplify the above formulation of $h(\mathbf{x}_i)$ to

$$h(\mathbf{x}_i) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x})$$



(Left:) The original data is 1-dimensional (top row) or 2-dimensional (bottom row). There is no hyper-plane that passes through the origin and separates the red and blue points.

(Right:) After a constant dimension was added to all data points, such a hyperplane exists.

## Hyperplane

**Observation**

Note that

$$y_i(\mathbf{w}^\top \mathbf{x}_i) > 0 \Longleftrightarrow \mathbf{x}_i \quad \text{is classified correctly}$$

where 'classified correctly' means that $x_i$ is on the correct side of the hyperplane defined by $\mathbf{w}$. Also, note that the left side depends on $y_i \in \{-1, +1\}$(it wouldn't work if, for example $y_i \in \{0, +1\}$).

## Table of Contents

## Algorithm

Now that we know what the **w** is supposed to do (defining a hyperplane the separates the data), let's look at how we can get such **w**.

```
Initialize $\vec{w} = \vec{0}$                    // Initialize $\vec{w}$. $\vec{w} = \vec{0}$ misclassifies everything.
while TRUE do                                      // Keep looping
    $m = 0$                                        // Count the number of misclassifications, $m$
    for $(x_i, y_i) \in D$ do                      // Loop over each (data, label) pair in the dataset, $D$
        if $y_i(\vec{w}^T \cdot \vec{x}_i) \leq 0$ then   // If the pair $(\vec{x}_i, y_i)$ is misclassified
            $\vec{w} \leftarrow \vec{w} + y\vec{x}$        // Update the weight vector $\vec{w}$
            $m \leftarrow m + 1$                   // Counter the number of misclassification
        end if
    end for
    if $m = 0$ then                                // If the most recent $\vec{w}$ gave 0 misclassifications
        break                                      // Break out of the while-loop
    end if
end while                                          // Otherwise, keep looping!
```

## Geometric Intuition

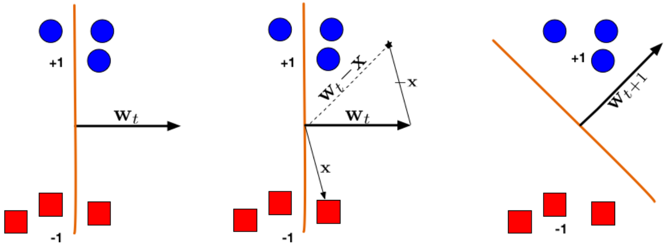


Illustration of a Perceptron update.(Left:) The hyperplane defined by $\mathbf{w}_t$ misclassifies one red (-1) and one blue (+1) point. (Middle:) The red point $\mathbf{x}$ is chosen and used for an update. Because its label is -1 we need to **subtractx** from $\mathbf{w}_t$. (Right:) The udpated hyperplane $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{x}$ separates the two classes and the Perceptron algorithm has converged.

## Geometric Intuition

**Quiz**

Assume a data set consists only of a single data point $\{(\mathbf{x}, +1)\}$. How often can a Perceptron misclassify this point $\mathbf{x}$ repeatedly?

What if the initial weight vector $\mathbf{w}$ was initialized randomly and not as the all-zero vector?

## Table of Contents

## Perceptron Convergence

The Perceptron was arguably the first algorithm with a strong formal guarantee. If a data set is linearly separable, the Perceptron will find a separating hyperplane in a finite number of updates. (If the data is not linearly separable, it will loop forever.)

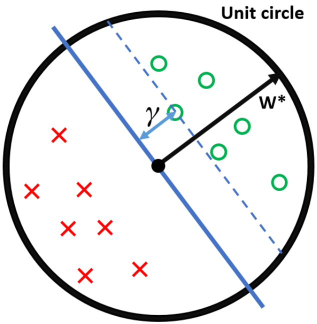
The argument goes as follows: Suppose $\exists \mathbf{w}^*$ such that $y_i(\mathbf{x}^\top \mathbf{w}^*) > 0 \ \forall (\mathbf{x}_i, y_i) \in D$

Now, suppose that we rescale each data point and the $\mathbf{w}^*$ such that

$$||\mathbf{w}^*|| = 1 \qquad \text{and} \qquad ||\mathbf{x}_i|| \leq 1 \quad \forall \mathbf{x}_i \in D$$

## Perceptron Convergence

Let us define the Margin $\gamma$ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{x}_i^\top \mathbf{w}^*|$.



To summarize our setup:

- All inputs $\mathbf{x}_i$ live within the unit sphere
- There exists a separating hyperplane defined by $\mathbf{w}^*$, with $\|\mathbf{w}\|^* = 1$ (i.e. $\mathbf{w}^*$ lies exactly on the unit sphere).
- $\gamma$ is the distance from this hyperplane (blue) to the closest data point.

## Theorem and Proof

**Theorem:** If all of the above holds, then the perceptron algorithm makes at most $1/\gamma^2$ mistakes.

**Proof:** Keeping what we defined above, consider the effect of an update ($\mathbf{w}$ becomes $\mathbf{w} + y\mathbf{x}$) on the two terms $\mathbf{w}^\top \mathbf{w}^*$ and $\mathbf{w}^\top \mathbf{w}$. We will use two facts:

- $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise we wouldn't make the update.
- $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

## Theorem and Proof

1.Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}^*$:

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + y(\mathbf{x}^\top \mathbf{w}^*) \geq \mathbf{w}^\top \mathbf{w}^* + \gamma$$

The inequality follows from the fact that, for $\mathbf{w}^*$, the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least $\gamma$ (i.e. $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$).
This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by **at least** $\gamma$.

2.Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}$:

$$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2(\mathbf{x}^\top \mathbf{x})}_{0\leq \quad \leq 1} \leq \mathbf{w}^\top \mathbf{w} + 1$$

The inequality follows from the fact that

- $2y(\mathbf{w}^\top \mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified
- $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$as $y^2 = 1$ and all $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$).

This means that for each update, $\mathbf{w}^\top \mathbf{w}$ grows by **at most 1**.

## Theorem and Proof

3.Now we can put together the above findings. Suppose we had $M$ updates.

$$\begin{aligned}
M\gamma &\leq \mathbf{w}^\top \mathbf{w}^* && \text{By first point} && (1)\\
&= |\mathbf{w}^\top \mathbf{w}^*| && \text{Simply because } M\gamma \geq 0 && (2)\\
&\leq ||\mathbf{w}||\ ||\mathbf{w}^*|| && \text{By Cauchy-Schwartz inequality}^* && (3)\\
&= ||\mathbf{w}|| && \text{As } ||\mathbf{w}^*|| = 1 && (4)\\
&= \sqrt{\mathbf{w}^\top \mathbf{w}} && \text{by definition of } \|\mathbf{w}\| && (5)\\
&\leq \sqrt{M} && \text{By second point} && (6)\\
& && && (7)\\
&\Rightarrow M\gamma \leq \sqrt{M} && && (8)\\
&\Rightarrow M^2\gamma^2 \leq M && && (9)\\
&\Rightarrow M \leq \frac{1}{\gamma^2} && && (10)
\end{aligned}$$

And hence, the number of updates $M$ is bounded from above by a constant.

* Alternative explanation: $|\mathbf{w}^\top \mathbf{w}^*| = \|\mathbf{w}\|\|\mathbf{w}^*\|| \cos(\alpha)|$, but $|\cos(\alpha)| \leq 1$

## Theorem and Proof

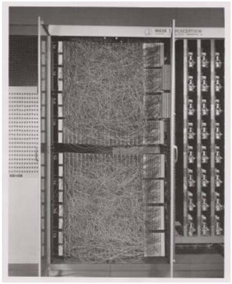
**Quiz**

Given the theorem above, what can you say about the margin of a classifier (what is more desirable, a large margin or a small margin?) Can you characterize data sets for which the perceptron algorithm will converge quickly? Draw an example.

## Table of Contents

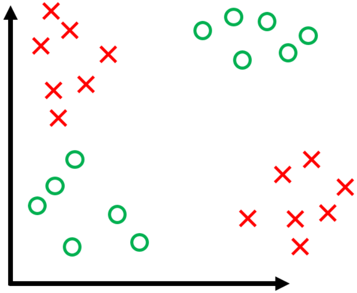
## Perceptron in the History

- The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by **Frank Rosenblatt**.
- Mark I Perceptron machine, the first implementation of the perceptron algorithm. It was connected to a camera with $20 \times 20$ cadmium sulfide photocells(光细胞阵列) to make a 400-pixel image. The main visible feature is a patch panel that set different combinations of input features. To the right, arrays of potentiometers(强力计阵列) that implemented the adaptive weights.

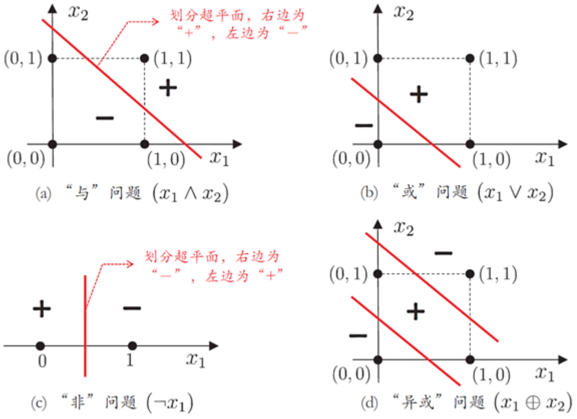


## Perceptron in the History

- Initially, huge wave of excitement ("Digital brains") (See The New Yorker December 1958)
- Then, contributed to the A.I. Winter. Famous example of a simple non-linearly separable data set, the XOR problem (Minsky 1969):



## AND, OR, NOT, XOR

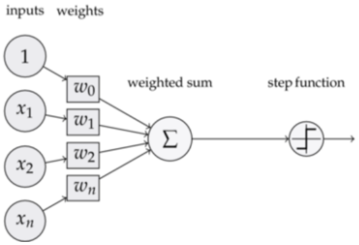


$x_2$
划分超平面，右边为“+”，左边为“−”
(0, 1)
(1, 1)
+
−
(0, 0)
(1, 0)
$x_1$
(a) “与”问题 $(x_1 \wedge x_2)$
$x_2$
(0, 1)
(1, 1)
+
−
(0, 0)
(1, 0)
$x_1$
(b) “或”问题 $(x_1 \vee x_2)$
划分超平面，右边为“−”，左边为“+”
+
−
0
1
$x_1$
(c) “非”问题 $(\neg x_1)$
$x_2$
−
(0, 1)
(1, 1)
+
−
(0, 0)
(1, 0)
$x_1$
(d) “异或”问题 $(x_1 \oplus x_2)$

图 5.4 线性可分的“与”“或”“非”问题与非线性可分的“异或”问题

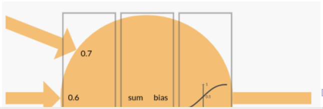
## Understanding the Perceptron as Neuron

- The “Mark 1 perceptron “ is machine was designed for image recognition: it had an array of 400 photocells(光细胞阵列), randomly connected to the “neurons”.
- Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors.



## From Perceptron to Neurual Network

- **Input**: All the feature becomes the input for a perceptron, $x = [x_1, x_2, ..., x_n]$.
- **Weights**: Weights are the values that are computed over the time of training the model. Initial we start the value of weights with some initial value and these values get updated for each training error. $w = [w_1, w_2, ...w_n]$.
- **BIAS**: A bias neuron allows a classifier to shift the decision boundary left or right. In an algebraic term, the bias neuron allows a classifier to translate its decision boundary. BIAS helps to training the model faster and with better quality.
- **Weighted Summation**: Weighted Summation is the sum of value that we get after the multiplication of each weight associated the each feature value.
- **Step/Activation Function**: the role of activation functions is make neural networks non-linear.
- **Output**: The weighted Summation is passed to the step/activation function and whatever value we get after computation is our predicted output.



## Table of Contents

## Reference

https://www.cs.cornell.edu/courses/cs4780/2018fa/page18/

The End