Part I

Shallow Model

CHAPTER 1

PERCEPTRON

1 The Structure

The perceptron is an algorithm for learning a binary classifier called a threshold function: a function that maps its input $\mathbf{x}$ (a real-valued vector) to an output value $f(\mathbf{x})$ (a single binary value):

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) \tag{1.1}$$

where $\sigma(z)$ is a function that outputs $+1$ if $z \geq 0$ and -1 otherwise, $\mathbf{w}$ is a vector of real-valued weights, $\mathbf{w}^T\mathbf{x}$ is the dot product

$$\mathbf{w}^T\mathbf{x} = \sum_{i=1}^{m} w_i x_i, \tag{1.2}$$

where m is the number of inputs to the perceptron, and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

Equivalently, since

$$\mathbf{w}^T\mathbf{x} + b = (\mathbf{w}, b) \cdot (\mathbf{x}, 1), \tag{1.3}$$

we can add the bias term b as another weight $\hat{\mathbf{w}}$ and add a coordinate 1 to each input $\hat{\mathbf{x}}$, and then write it as a linear classifier that passes through the origin:

$$f(\mathbf{x}) = \sigma(\hat{\mathbf{w}}^T\hat{\mathbf{x}}) \tag{1.4}$$

The binary value of $f(\mathbf{x})$ (0 or 1) is used to perform binary classification on $\mathbf{x}$ as either a positive or a negative instance. Spatially, the bias shifts the position (though not the orientation) of the planar decision boundary.

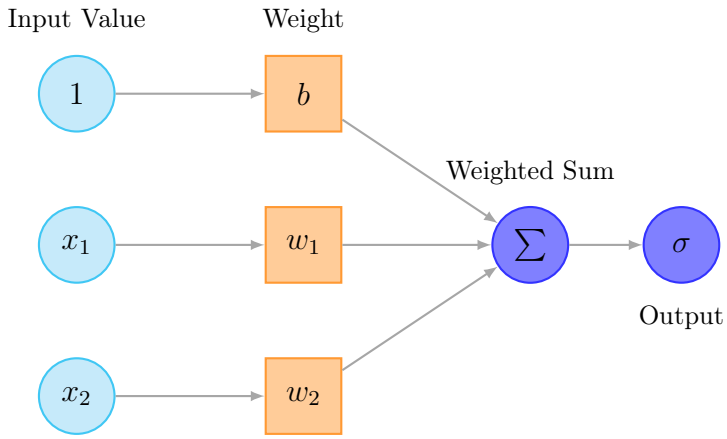

Figure 1.1: The perceptron structure

2 XOR Problem

Theorem 2.1: Linear Separable

If 3 points are collinear and the middle point has a different label than the other two, then these 3 points cannot be linearly separable.

Proof for Theorem.

Let us label the points as point A, B, and C. A and C have the same label, and B has a different label. They are all collinear with line $\mathcal{L}$.

Assume a contradiction: a line can linearly separate A, B, and C. This means a line must cross between segment AB and segment BC to linearly separate these three points (by the definition of linear separability). Let us label the point where the line crosses segment AB as point Y, and the point where the line crosses segment BC as point Z.

However, since segments AB and BC are collinear with line $\mathcal{L}$, points Y and Z also fall on line $\mathcal{L}$. Since only one unique line can pass through two points, it must be that the only line that passes through segments AB and BC (and therefore separates points A, B, and C) is line $\mathcal{L}$.

However, line $\mathcal{L}$ cannot linearly separate A, B, and C, since line $\mathcal{L}$ also passes through them. Therefore, no line exists that can separate A, B, and C. ∎

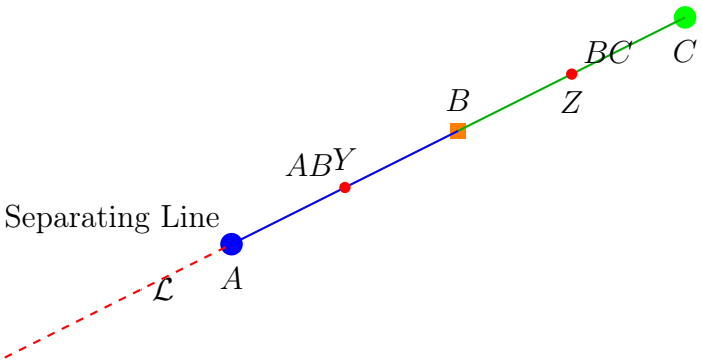

Figure 1.2: The linear separable problem

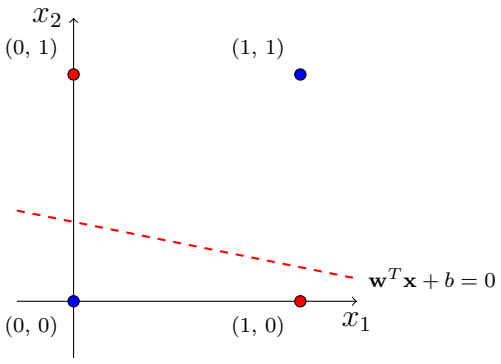

Figure 1.3: The perceptron can not solve XOR problems.

Theorem 2.2: Perceptron can not solve XOR problem

The XOR problem involves the following input-output pairs:

x_1	x_2	$y = \text{XOR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Here, y represents the XOR output. The task is to find $\mathbf{w}$ and b such that the perceptron correctly classifies all four input-output pairs.

Proof for Theorem.

A perceptron can solve a problem if and only if the data points are **linearly separable**. This means there exists a hyperplane:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

that separates the two classes $y = 0$ and $y = 1$.

Assume, for contradiction, that there exists a perceptron with weights $\mathbf{w}$ and bias b that solves the XOR problem. Then the perceptron must satisfy the following inequalities:

1. $b < 0$ (for $(x_1, x_2) = (0, 0)$, $y = 0$),
2. $w_2 + b > 0$ (for $(x_1, x_2) = (0, 1)$, $y = 1$),
3. $w_1 + b > 0$ (for $(x_1, x_2) = (1, 0)$, $y = 1$),
4. $w_1 + w_2 + b < 0$ (for $(x_1, x_2) = (1, 1)$, $y = 0$).

Adding inequalities 2 and 3:

$$(w_2 + b) + (w_1 + b) > 0 \implies w_1 + w_2 + 2b > 0.$$

But inequality 4 states:

$$w_1 + w_2 + b < 0.$$

This leads to a contradiction, because $w_1 + w_2 + 2b > 0$ cannot simultaneously satisfy $w_1 + w_2 + b < 0$ with $b < 0$. ∎

2.1 Learning Algorithm

Algorithm 2.1: Perceptron Learning Algorithm

Input: Training set $\{(\hat{\mathbf{x}}_i, y_i)\}_{i=1}^n$, learning rate η
Initialize: $\hat{\mathbf{w}}_1 = 0$
for t = 1 to T **do**
  for each training sample $(\hat{\mathbf{x}}_i, y_i)$ **do**
    if the sample is misclassified **then**
      $\hat{\mathbf{w}}_{t+1} = \hat{\mathbf{w}}_t + \eta y_i \hat{\mathbf{x}}_i$
    else
      leave $\hat{\mathbf{w}}_{t+1}$ unchanged
    end if
  end for
end for
Output: Weight vector $\hat{\mathbf{w}}_T$

Theorem 2.3: Bound on the number of errors in the Perceptron algorithm in Algorithm 2.1

Assume that there exists some parameter vector $\mathbf{w}^*$ such that $\|\mathbf{w}^*\| = 1$, and some $\gamma > 0$ such that for all $t = 1, \ldots, n$:

$$y_t(\mathbf{x}_t^\top \mathbf{w}^*) \geq \gamma$$

Assume in addition that for all $t = 1, \ldots, n$, $\|\mathbf{x}_t\| \leq R$. Under these assumptions, the Perceptron algorithm makes at most:

$$\frac{R^2}{\gamma^2}$$

errors.
Definition of an error: An error occurs whenever we have $f(\mathbf{x_t}) \neq y_t$ for some (j, t) pair in the algorithm.

Proof for Theorem.

To show that the number of errors is bounded, we define $\mathbf{w}_k$ as the weight vector after the k-th error. Initially, we have $\mathbf{w}_1 = 0$. When the algorithm makes an error on an example $(\mathbf{x}_t, y_t)$, the update rule is:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + y_t \mathbf{x}_t$$

We then calculate the inner product $\mathbf{w}_k^\top \mathbf{w}^*$, where $\mathbf{w}^*$ is the optimal separating hyperplane. Since the Perceptron algorithm updates the weight vector in the direction of the misclassified example, we have:

$$\mathbf{w}_{k+1}^\top \mathbf{w}^* = (\mathbf{w}_k + y_t \mathbf{x}_t)^\top \mathbf{w}^* = \mathbf{w}_k^\top \mathbf{w}^* + y_t(\mathbf{x}_t^\top \mathbf{w}^*) \geq \mathbf{w}_k^\top \mathbf{w}^* + \gamma$$

where γ is the margin. By induction, after k updates, we have:

$$\mathbf{w}_k^\top \mathbf{w}^* \geq k\gamma$$

Next, we bound the squared norm of $\mathbf{w}_k$. From the update rule, we have:

$$\|\mathbf{w}_{k+1}\|^2 = \|\mathbf{w}_k\|^2 + 2y_t(\mathbf{w}_k^\top \mathbf{x}_t) + \|\mathbf{x}_t\|^2$$

Assume $\|\mathbf{x}_t\| \leq R$, we obtain:

$$\|\mathbf{w}_{k+1}\|^2 = \|\mathbf{w}_k + y_t \mathbf{x}_t\|^2 = \|\mathbf{w}_k\|^2 + y_t^2 \|\mathbf{x}_t\|^2 + 2y_t(\mathbf{w}_k^\top \mathbf{x}_t) \leq \|\mathbf{w}_k\|^2 + R^2$$

The inequality follows because: 1) $y_t^2 \|\mathbf{x}_t\|^2 = \|\mathbf{x}_t\|^2 \leq R^2$, as assumed, and because $y_t^2 = 1$. 2) $y_t(\mathbf{w}_k^\top \mathbf{x}_t) \leq 0$, since the parameter vector $\mathbf{w}_k$ gave an error on the t-th example.

It follows by induction on k (recall that $\|\mathbf{w}_1\|^2 = 0$), that:

$$\|\mathbf{w}_{k+1}\|^2 \leq kR^2$$

Combining the lower and upper bounds, we have:

$$k^2\gamma^2 \leq \|\mathbf{w}_{k+1}\|^2 \leq kR^2$$

from which it follows that:

$$k \leq \frac{R^2}{\gamma^2}.$$

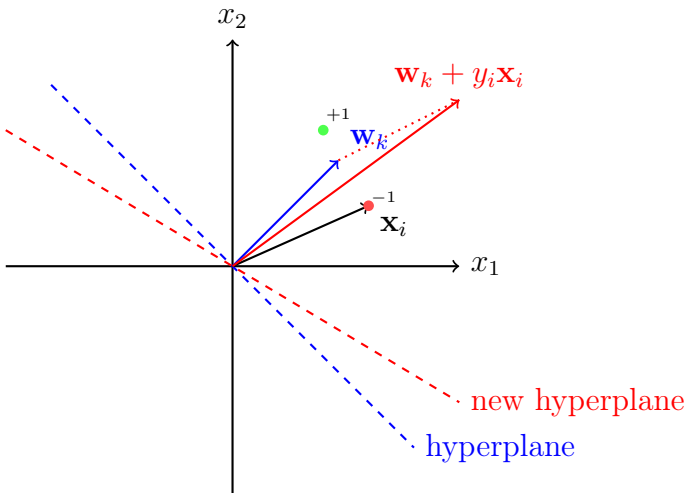

Figure 1.4: This figure illustrates the perceptron update process. The blue dashed line represents the initial hyperplane, which is perpendicular to the weight vector $\mathbf{w}_k$ (blue arrow). The red dashed line represents the new hyperplane after the weight vector is updated to $\mathbf{w}_k + y_i\mathbf{x}_i$ (red arrow), which rotates the hyperplane according to the position of $\mathbf{x}_i$. A misclassified sample $\mathbf{x}_i$ (red point) causes this update, while a correctly classified sample (green point) remains unaffected.