Rosenblatt's Perceptron: The Basis of Neural Networks

Oliver Zhao

1 Introduction

Rosenblatt's perceptron - also just called a perceptron - is a supervised binary classifier in which the input is represented by a vector of numbers. It is used to classify patterns that are linearly separable. The perceptron is the building block of neural networks. Through the *Perceptron Convergence Theorem*, we can prove that the perceptron algorithm converges and positions a hyperplane between the two classes of data, provided the data is linearly separable.

2 What the Perceptron Does

Rosenblatt's perceptron takes in a vector of m inputs $\mathbf{x} = \{x_0, x_1, \cdots, x_m\}$, and gives out a single output y. For each of these inputs, there is an associated weight from the weight vector $\mathbf{w} = \{w_0, w_1, \cdots, w_m\}$, where weight w_i corresponds to input x_i. Note that we will use the convention $x_0 = +1$, where b is the bias with weight $w_0 = b$. These inputs and bias are summed together by their weights, to give a final input v that is fed into Rosenblatt's perceptron,

$$v = \sum_{i=0}^{m} w_i x_i = \sum_{i=1}^{m} w_i x_i + b. \tag{2.1}$$

This input value v is fed into a sign function $\phi(\cdot)$, where

$$\phi(v) = sgn(v) = \begin{cases} 1 & v > 0 \\ -1 & v \leq 0 \end{cases}. \tag{2.2}$$

The perceptron separates two class regions by a hyperplane, where outputs greater than the hyperplane is defined as class $\mathscr{C}_1$ and outputs less than the

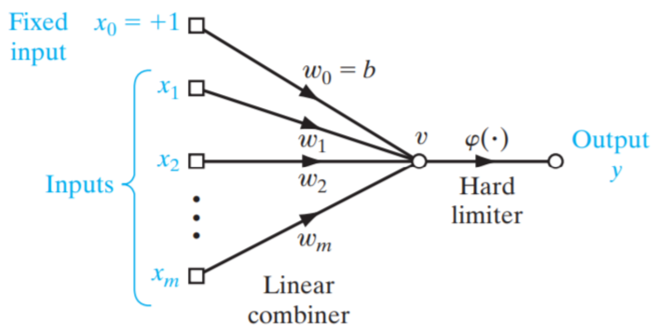

Figure 1: Diagram of Rosenblatt's perceptron.

hyperplane is defined as class $\mathscr{C}_2$. The hyperplane is defined by

$$\sum_{i=1}^{m} w_i x_i + b = 0. \tag{2.3}$$

For example, consider the figure below, where we see a two-dimensional, two-class classification problem. We see that excluding the bias $b = w_0 x_0 = x_0$, there are only two inputs (hence a two-dimensional problem).

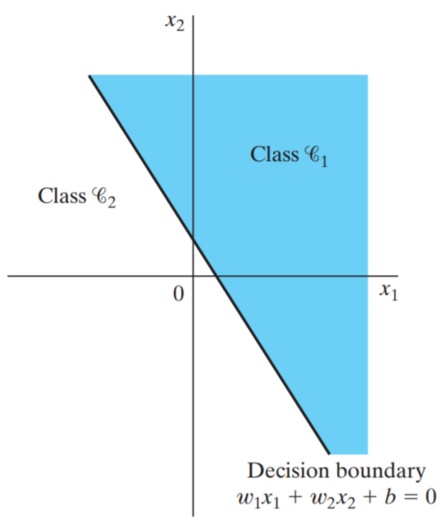

Figure 2: Decision boundary separating $\mathscr{C}_1$ and $\mathscr{C}_2$.

3 How the Perceptron Works

Now we will work to derive the error-correction learning algorithm for Rosenblatt's perceptron. We have a $(m+1)$-by-1 input vector, where n denotes the iteration in applying the algorithm

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \cdots, x_m(n)]^T. \tag{3.1}$$

The corresponding $(m+1)$-by-1 weight vector is

$$\mathbf{w}(n) = [b, w_1(n), w_2(n), \cdots, w_m(n)]^T. \tag{3.2}$$

Notice that the bias b is incorporated into the input vector and weight vector as $w_0 x_0 = b$. Now, the final input can be written more concisely as

$$v(n) = \sum_{i=0}^{m} w_i(n) x_i(n) = \mathbf{w}^T(n)\mathbf{x}(n). \tag{3.3}$$

Note that in order for this perceptron to succeed in classification, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ must be linearly separable. Now, we can state that there exists a weight vector w such that the following is true

$$\begin{aligned} &\mathbf{w}^T\mathbf{x} > 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_1 \\ &\mathbf{w}^T\mathbf{x} \leq 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_2 \end{aligned} \tag{3.4}$$

The goal of the model is to find this ideal weight vector w that satisfies the conditions above. If the nth member of the training set $\mathbf{x}(n)$ is classified correctly by the weight vector $\mathbf{w}(n)$ at the nth iteration of the algorithm by the weight vector $\mathbf{w}(n)$ at the nth iteration of the algorithm, then no correction is made to the weight vector at the perceptron. This can be mathematically stated as

$$\begin{aligned} &\mathbf{w}(n+1) = \mathbf{w}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ &\mathbf{w}(n+1) = \mathbf{w}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{aligned} \tag{3.5}$$

If the nth member of the training set $\mathbf{x}(n)$ is classified *incorrectly* by the weight vector $\mathbf{w}(n)$ at the nth iteration of the algorithm, the the weight vector of the perceptron is updated with the following rule

$$\begin{aligned} &\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \\ &\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \text{ if } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \end{aligned} \tag{3.6}$$

Where $\eta(n)$ is the *learning-rate parameter* that controls how extreme the weight vector adjustments are. If $\eta(n) = \eta > 0$, then the learning-rate parameter is constant through every iteration, known as *fixed-increment adaptation*. In neural networks, the learning-rate parameter is often not a fixed value.

4 Perceptron Convergence Algorithm

Variables and Parameters

$$
\begin{aligned}
\mathbf{x}(n) &= (m+1)\text{-by-1 input vector} \\
&= [+1, x_1(n), x_2(n), c\ldots, x_m(n)]^T \\
\mathbf{w}(n) &= (m+1)\text{-by-1 weight vector} \\
&= [b, w_1(n), w_2(n), c\ldots, w_m(n)]^T \\
b &= \text{bias} \\
y(n) &= \text{actual response} \\
d(n) &= \text{desired response} \\
\eta &= \text{learning-rate parameter } (0 < \eta < 1)
\end{aligned}
$$

$d(n)$ can be viewed as the "true" class of the data point, and hence the desired response, defined by

$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases} \tag{4.1}$$

Meanwhile, $y(n)$ is the predicted class of the data point, calculated as

$$y(n) = sgn[\mathbf{w}^T(n)\mathbf{x}(n)]. \tag{4.2}$$

In order to adapt the weight vector, we update it through the following rule

$$\mathbf{w}(n+1) = \mathbf{w}(n) = \eta[d(n) - y(n)]\mathbf{x}(n). \tag{4.3}$$

Note that the difference $d(n) - y(n)$ is what we call the *error signal*.

Algorithm

1. *Initialization.* Set $\mathbf{w}(0) = (0)$. Then perform the following computations for iterations $n = 1, 2, \ldots,$ until converging on a solution.

2. *Activation.* At iteration n activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.

3. *Computation of Actual Response.* Compute the actual response of the perceptron as
$$y(n) = sgn[\mathbf{w}^T(n)\mathbf{x}(n)] \tag{4.4}$$

4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain
$$\mathbf{w}(n+1) = \mathbf{w}(n) = \eta[d(n) - y(n)]\mathbf{x}(n) \tag{4.5}$$

5. *Continuation.* Increment iteration n by one and go back to step 2.

5 Proof of Perceptron Convergence Theorem

We first the convergence of a fixed-increment adaptation rule for which $\eta = 1$. Consider the initial condition $\mathbf{w}(0) = \mathbf{0}$. Suppose that $\mathbf{w}^T(n)\mathbf{x}(n) < 0$ for $n = 1, 2, \ldots$, and the input vector $\mathbf{x}(n)$ belongs to the subset $\mathscr{H}_1$. That is, the perceptron incorrectly classifies the vectors $\mathbf{x}(1), \mathbf{x}_2, \ldots$. Then with the constant $\eta(n) = 1$, we may write

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n) \text{ for } \mathbf{x}(n) \quad \text{belonging to class } \mathscr{L}_1. \tag{5.1}$$

Given the initial condition $\mathbf{w}(0) = \mathbf{0}$, we may iteratively solve this equation for $\mathbf{w}(n+1)$, obtaining

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(2) + \cdots + \mathbf{x}(n). \tag{5.2}$$

Since the classes $\mathscr{L}_1$ and $\mathscr{L}_2$ are assumed to be linearly separable, there exists a solution $\mathbf{w}_o$ for which $\mathbf{w}^T\mathbf{x}(n) > 0$ for the vectors $\mathbf{x}(1), \ldots, \mathbf{x}(n)$ belonging to the subset $\mathscr{H}_1$. For a fixed solution $\mathbf{w}_o$, we may then define a positive number α as

$$\alpha = \min_{\mathbf{x}(n) \epsilon \mathscr{H}_1} \mathbf{w}_o^T \mathbf{x}(n). \tag{5.3}$$

Hence, multiplying both sides of Equation (5.2) by the row vector $\mathbf{w}_o^T$, we get

$$\mathbf{w}_o^T\mathbf{w}(n+1) = \mathbf{w}_o^T\mathbf{x}(1) + \mathbf{w}_o^T\mathbf{x}(2) + \cdots + \mathbf{w}_o^T\mathbf{x}(n). \tag{5.4}$$

In light of the definition given in Equation (5.3), we have

$$\mathbf{w}_o^T\mathbf{w}(n+1) \geq n\alpha. \tag{5.5}$$

Given two vectors $\mathbf{w}_0$ and $\mathbf{w}(n+1)$, the Cauchy-Schwarz inequality states that

$$||\mathbf{w}_o||^2||\mathbf{w}(n+1)||^2 \geq [\mathbf{w}_o^T\mathbf{w}(n+1)]^2. \tag{5.6}$$

We now note from Equation 5.5 that $[\mathbf{w}_o^T\mathbf{w}(n+1)]^2$ is equal to or greater than $n^2\alpha^2$. From Equation (5.6), it follows that

$$||\mathbf{w}_o||^2||\mathbf{w}(n+1)||^2 \geq n^2\alpha^2, \tag{5.7}$$

or

$$||\mathbf{w}(n+1)||^2 \geq \frac{n^2\alpha^2}{||\mathbf{w}_o||^2}. \tag{5.8}$$

Now let us rewrite Equation (5.1) as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k) \quad \text{for} \quad k = 1, \ldots, n \quad \text{and} \quad \mathbf{x}(k) \; \epsilon \; \mathscr{H}_1. \tag{5.9}$$

By taking the squared Euclidean norm of both sides of Equation (5.9), we get

$$||\mathbf{w}(k+1)||^2 = ||\mathbf{w}(k)||^2 + ||\mathbf{x}(k)||^2 + 2\mathbf{w}^T(k)\mathbf{x}(k). \tag{5.10}$$

But, $\mathbf{w}^T(k)\mathbf{x}(k) \leq 0$. Therefore, we can deduce from Equation (5.10) that

$$||\mathbf{w}(k+1)||^2 \leq ||\mathbf{w}(k)||^2 + ||\mathbf{x}(k)||^2 \tag{5.11}$$

or

$$||\mathbf{w}(k+1)||^2 - ||\mathbf{w}(k)||^2 \leq ||\mathbf{x}(k)||^2, \quad k = 1, \ldots, n. \tag{5.12}$$

Adding these inequalities for $k = 1, \ldots, n$ and invoking the assumed initial condition $\mathbf{w}(0) = \mathbf{0}$, we get the inequality

$$\begin{aligned} ||\mathbf{w}(n+1)||^2 &\leq \sum_{k=1}^{n} ||\mathbf{x}(k)||^2 \\ &\leq n\beta \end{aligned} \tag{5.13}$$

Where β is a positive number defined as

$$\beta = \max_{\mathbf{x}(k) \epsilon \mathscr{H}_1} ||\mathbf{x}(k)||^2. \tag{5.14}$$

Equation (5.13) states that the squared Euclidean norm of the weight vector $\mathbf{w}(n+1)$ grows at most linearly with the number of iterations n. The second result of (5.13) is in conflict with the result of Equation (5.8) for sufficiently large values of n. We can state that n cannot be larger than some value $n_{\max}$

for which Equations (5.8) and (5.13) are both satisfied with the equality sign. That is, $n_{\max}$ is the solution to the equation

$$\frac{n_{\max}^2 \alpha^2}{||\mathbf{w}_o||^2} = n_{\max}\beta. \tag{5.15}$$

Solving for $n_{\max}$ given a solution vector $\mathbf{w}_o$, we find that

$$n_{\max} = \frac{\beta||\mathbf{w}_o||^2}{\alpha^2}. \tag{5.16}$$

Thus, we proved that for $\eta(n) = 1$ for all n and $\mathbf{w}(0) = \mathbf{0}$, and given that a solution vector $\mathbf{w}_o$ exists, the rule for adapting the synaptic weights of the perceptron must terminate after at most $n_{\max}$ iterations. This statement, proved for hypothesis $\mathscr{H}_1$, also holds for hypothesis $\mathscr{H}_2$.