Computational (AI) Models

John Sum
Institute of Technology Management
National Chung Hsing University
Taichung 402, Taiwan

March 26, 2025

Abstract

One should be noted that an AI model is in essence a computational model. Given an input vector with numbers, the AI model computes the output vector with numbers. In this article, the key concepts on computational models are presented with introducing a few computational AI models. Perceptron learning rule and backpropagation learning rule are introduced. Simulation results are shown to illustrate the properties of those learning rules which are associated with their computational models, namely simple Perceptron and multilayered Perceptron (MLP). Furthermore, recurrent networks are introduced and highlighted their potential applications in problems regarding sequence generations.

Contents

List of Figures

List of Tables

1 McCulloch-Pitts Neuronal Networks

McCulloch-Pitts model is the first mathematical model proposed by W. McCulloch and W. Pitts in 1943 [1]. This model abstracts the *all-or-none* property of a neuron – *If the stimulus feeding to a neuron is larger enough, the neuron fires.*

1.1 McCulloch-Pitts Neuron Model

Consider a neuron with n inputs and let $x_1, \cdots, x_n$ be the inputs. $x_i \in \{0, 1\}$ for $i = 1, \cdots, n$. Let $y = f(\mathbf{x})$ be the neuron output. The output is defined as follows :

$$f(\mathbf{x}) = h\left(\sum_{i=1}^{n} w_i x_i - b\right), \tag{1}$$

where

$$h(u) = \begin{cases} 1 & \text{if } u > 0, \\ 0 & \text{if } u \leq 0. \end{cases} \tag{2}$$

1.2 2-Input-1-Output M-P Neuron

Figure 1 shows a model with two inputs. In the figure, w_1 and w_2 are called the synaptic weights. They act like scaling factors controlling the effects of the inputs to the neuron. b is called the threshold (or bias). If the weighted sum of the inputs is larger than the threshold b, the neuron fires (equivalently, $f(\mathbf{x}) = 1$. The $h(\cdot)$ in the neuron is a step function as defined in (2).

One should be noted that the inputs are all binary variables which are non-negatives. If the value of w_1 (resp. w_2) is positive, the effect of x_1 (resp. x_2) to the neuron is *excitatory*. If the value of w_1 (resp. w_2) is negative, the effect of x_1 (resp. x_2) to the neuron is *inhibitory*.

1.3 Geometrical Interpretation

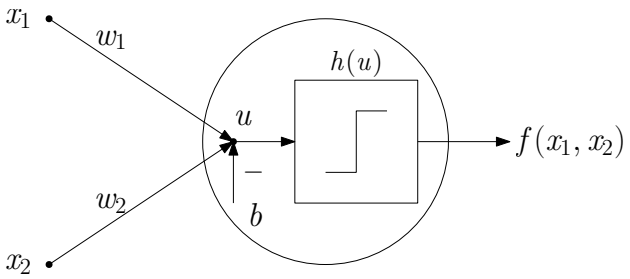
For a two-input-one-output neuron as shown in Figure 1, its mathematical model can simply be given by

$$f(x_1, x_2) = h(\underbrace{w_1 x_1 + w_2 x_2 - b}_{u(x_1, x_2)}). \tag{3}$$

Here, the function $u(x_1, x_2)$ inside $h(\cdot)$ is called a decision boundary. It partitions a 2-D plane into two parts. On one side of the decision boundary, $h(u(x_1, x_2)) > 1$. On the other side, $h(u(x_1, x_2)) < 0$.

Figure 2 shows the two examples, in which

$$\begin{aligned} u(x_1, x_2) &= x_1 + x_2 - 1. \\ u(x_1, x_2) &= -2x_1 + \frac{8}{3} x_2 - 1. \end{aligned}$$



$$f(x_1, x_2) = h(\underbrace{w_1 x_1 + w_2 x_2 - b}_{u}).$$

$$h(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{if } u \leq 0. \end{cases}$$

Figure 1: A McCulloch-Pitts model of a neuron with two inputs. Here, w_1 and w_2 are the synaptic weights; b is called the bias and $h(\cdot)$ is a step function. If the value of w_1 (resp. w_2) is positive, the effect of x_1 (resp. x_2) to the neuron is excitatory. If the value of w_1 (resp. w_2) is negative, the effect of x_1 (resp. x_2) to the neuron is inhibitory.

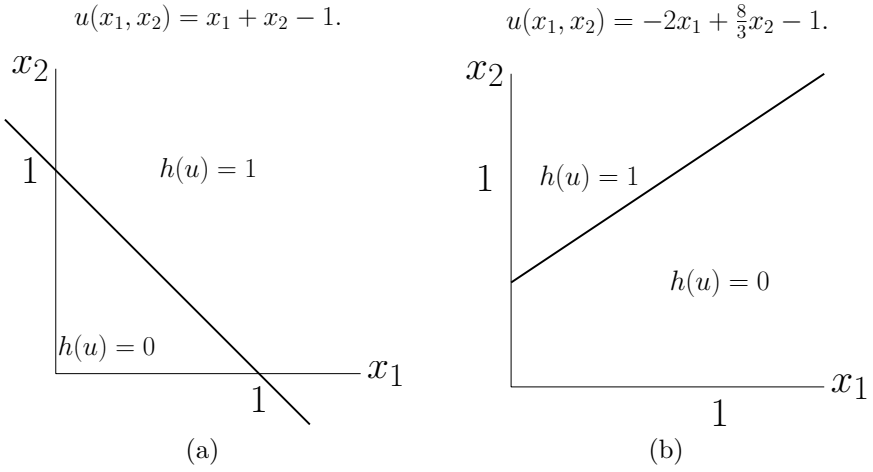
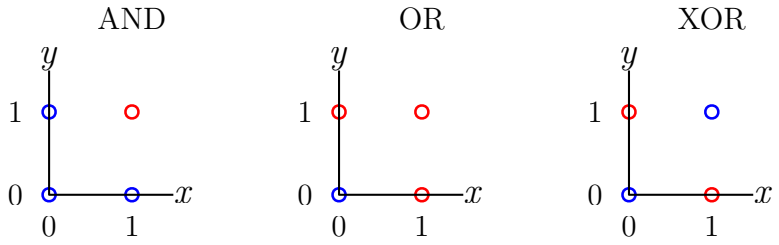


Figure 2: Two exemplar decision boundaries. (a) $u(x_1, x_2) = x_1 + x_2 - 1$. (b) $u(x_1, x_2) = -2x_1 + 8x_2/3 - 1$.



AND

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Figure 3: Geometrical interpretation of three logical operations, namely logical AND (left), logical OR (middle) and XOR (right), and their truth tables.

1.4 Logical Operations Realization

For a single two-input McCulloch-Pitts neuron with specified values for w_1, w_2 and b, one can use the neuron to perform some logical operations. Figure 3 shows the geometries of three logical operations and their truth tables.

1.4.1 AND: $w_1 = w_2 = 1$, $b = 1.5$

For $w_1 = w_2 = 1$, $b = 1.5$, the neuronal model is given by

$$f(x_1, x_2) = h(x_1 + x_2 - 1.5). \tag{4}$$

As $x_1, x_2 \in \{0, 1\}$, $f(x_1, x_2) = 1$ if and only if $x_1 = x_2 = 1$. The neuron as defined by (4) performs logical AND. It acts as an AND gate.

1.4.2 OR: $w_1 = w_2 = 1$, $b = 0.5$

For $w_1 = w_2 = 1$, $b = 0.5$, the neuronal model is given by

$$f(x_1, x_2) = h(x_1 + x_2 - 0.5). \tag{5}$$

As $x_1, x_2 \in \{0, 1\}$, $f(x_1, x_2) = 0$ if and only if $x_1 = x_2 = 0$. The neuron as defined by (5) performs logical OR. It acts as an OR gate.

1.4.3 NAND: $w_1 = w_2 = -1$, $b = -1.5$

For $w_1 = w_2 = -1$, $b = -1.5$, the neuronal model is given by

$$f(x_1, x_2) = h(-x_1 - x_2 + 1.5). \tag{6}$$

As $x_1, x_2 \in \{0, 1\}$, $f(x_1, x_2) = 0$ if and only if $x_1 = x_2 = 1$. The neuron as defined by (6) performs logical NAND. It acts as an NAND gate.

1.4.4 NOR: $w_1 = w_2 = -1$, $b = -0.5$

For $w_1 = w_2 = -1$, $b = -0.5$, the neuronal model is given by

$$f(x_1, x_2) = h(-x_1 - x_2 + 0.5). \tag{7}$$

As $x_1, x_2 \in \{0, 1\}$, $f(x_1, x_2) = 1$ if and only if $x_1 = x_2 = 0$. The neuron as defined by (7) performs logical NOR. It acts as an NOR gate.

1.4.5 XOR Operation

For the above logical operations, their successes rely on proper designs of their decision boundaries given by

$$w_1 x_1 + w_2 x_2 - b = 0. \tag{8}$$

For each of the above logical operations, only one decision boundary is needed. As highlighted in [2], a single two-input McCulloch-Pitts neuron is unable to perform XOR operation. To do so, three two-input McCulloch-Pitts neurons are needed.

Figure 4 shows the network of three neurons which performs the XOR operation. Two neurons are needed in the (so-called) hidden layer. The outputs of the hidden neurons feed their output to the output neuron. The neurons in the hidden layer are defined to perform logical OR and logical NAND. Let $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ be the outputs of the hidden neurons. By (4), (5) and (6), we get that

$$\begin{aligned} f_1(x_1, x_2) &= h(x_1 + x_2 - 0.5), \\ f_2(x_1, x_2) &= h(-x_1 - x_2 + 1.5), \\ f(x_1, x_2) &= h(x_1 + x_2 - 1.5). \end{aligned}$$

That is to say, with the settings of $w_{11} = w_{12} = 1$, $b_1 = 0.5$, $w_{21} = w_{22} = -1$, $b_2 = -1.5$, $\alpha_1 = \alpha_2 = 1$, $\beta = 1.5$ for the three two-input McCulloch-Pitts neurons as shown in Figure 4, XOR can be implemented.

1.5 Network of McCulloch-Pitts Neurons

To go beyond, one can claim that all multiple-input-multiple-output binary system can be implemented by a network of two-input McCulloch-Pitts neurons. In view of the processing in each neuron, these networks are basically computational models. Given an input $\mathbf{x}$, the network simply computes the outputs in accordance with the computations of the neurons in the network. A network of two-input McCulloch-Pitts neurons is essentially a *computational model*. Precisely, it is a *multiple-binary-input-multiple-binary-output computational model*[1].

[1] Note that this model is a special class of models. For the input (resp. output) value is not limited to binary, the model is simply called multiple-input-multiple-output (MIMO) model

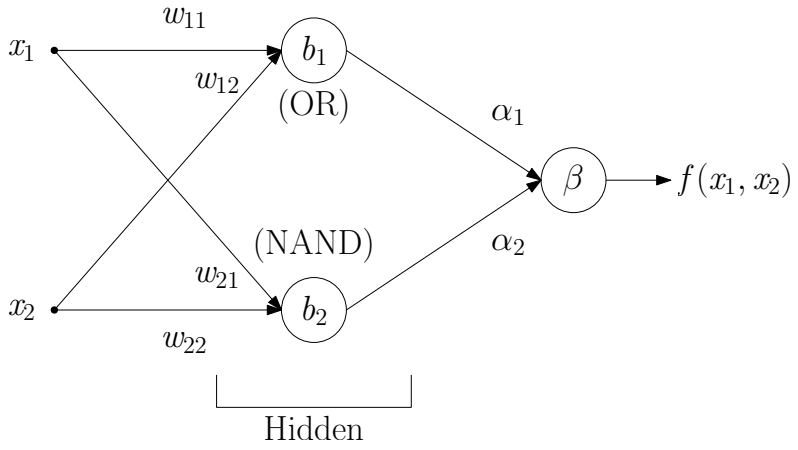


Figure 4: A network of three two-input McCulloch-Pitts neurons performs XOR operation. The neurons in the hidden layer are defined to perform logical OR and logical NAND. The parameters of the model are set to be $w_{11} = w_{12} = 1$, $b_1 = 0.5$, $w_{21} = w_{22} = -1$, $b_2 = -1.5$, $\alpha_1 = \alpha_2 = 1$, $\beta = 1.5$

1.5.1 Decision Network

To play Tic-Tac-Toe, one needs to block the opponent to fill up a line. If a line has already filled up with two opponent symbols, we should fill in the reminding un-filled cell with our symbol. To make this decision, Figure 5 shows a network of McCulloch-Pitts neurons for this decision making – *Should a symbol be put on the cell corresponding to* x_1, x_2 *or* x_3*?*

1.5.2 3-Input-4-Output Neuronal Network

To accomplish this, an AI model with three inputs and four outputs can be designed. In it, there are even types of neurons. Some of them are single-input-single-output neurons (a-type and b-type). Some of them are two-input-single-output neurons (f-type and g-type) and some of them are three-input-single-output neurons (c-type, d-type and e-type). Their mathematical models are given as follows :

$$f_a(x_i) = h(-x_i - 0.5), \tag{9}$$
$$f_b(x_i) = h(x_i - 0.5), \tag{10}$$
$$f_c(y_1, y_2, y_3) = h(y_1 + y_2 + y_3 - 1.5), \tag{11}$$
$$f_d(y_1, y_2, y_3) = h(-y_1 - y_2 - y_3 + 2.5), \tag{12}$$
$$f_e(y_1, y_2, y_3) = h(y_1 + y_2 + y_3 - 1.5), \tag{13}$$
$$f_f(f_c, f_d) = h(f_c + f_d - 1.5), \tag{14}$$
$$f_g(f_f, z_i) = h(f_e - z_i - 0.5), \tag{15}$$

(equi. system).

for $i = 1, 2, 3$. The outputs are defined as follows :

$$o_1 = f_g(f_f, z_1), \;\; o_2 = f_g(f_f, z_2), \;\; o_3 = f_g(f_f, z_3), \;\; o_4 = f_e(y_1, y_2, y_3). \quad (16)$$

If $o_i = 1$, fill in the cell x_i with a symbol. If $o_4 = 1$, the game is over.

While there are one-input-one-output neurons and two-input-one-output neurons in this network, we can replace them by using three-input-one-output neurons. The idea is straight forward. To implement an one-input-one-output neuron, we can set the weights of two inputs to zeros as shown in Figure 6. This idea can be extended to N-input-one-output neuron. The network as shown in Figure 5 can be implemented as a multilayered network as shown in Figure 7. With proper design on the values for the weights and biases, this network is able to (but not limited to) replicate the functionalities of the network as shown in Figure 5.

1.5.3 9-Input-10-Output Neuronal Network

Note that there are eight lines to be diagnosed – three rows, three columns and two diagonals. Decision on the next move in a tic-tac-toe game can be solved by a (bigger) network consisting of nine inputs and ten outputs. This big network is basically a consolidation of eight of the above network of McCulloch-Pitts neurons. Each network makes decision on the next possible move for a line.

1.6 Model Complexity

The complexity of a neuronal model is normally determined by the number of neurons and the number of parameters in the model. For comparison, the number of neurons and the number of parameters in the neuronal networks presented above are depicted in Table 1.

For the model denoted as $(N_0 - N_1 - \cdots - N_L)$ in Table 1 is a multilayered network with N_0 inputs and N_L outputs. $N_1, \cdots, N_{L-1}$ are the number of neurons in each layers. The total number of neurons is clearly $\sum_{k=1}^{L} N_k$ and the total number of parameters is $\sum_{k=1}^{N} N_k \left(N_{k-1} + 1\right)$. In which, $N_k N_{k-1}$ is the number of connections (i.e. parameters) between the k-layer neurons and the $(k-1)$-layer neurons.

1.7 Digital Computer and M-P Networks

Note that a computer is essentially constructed by a network of AND, OR, NAND, NOR and XOR logic gates to perform both logical and arithmetics operations. As a network of two-input McCulloch-Pitts neurons can perform the operations as the logic gates, a digital computer can thus be implemented by these two-input McCulloch-Pitts neurons. In this regard, a connection between computer and brain was established. *A human brain can do more than a digital computer.*

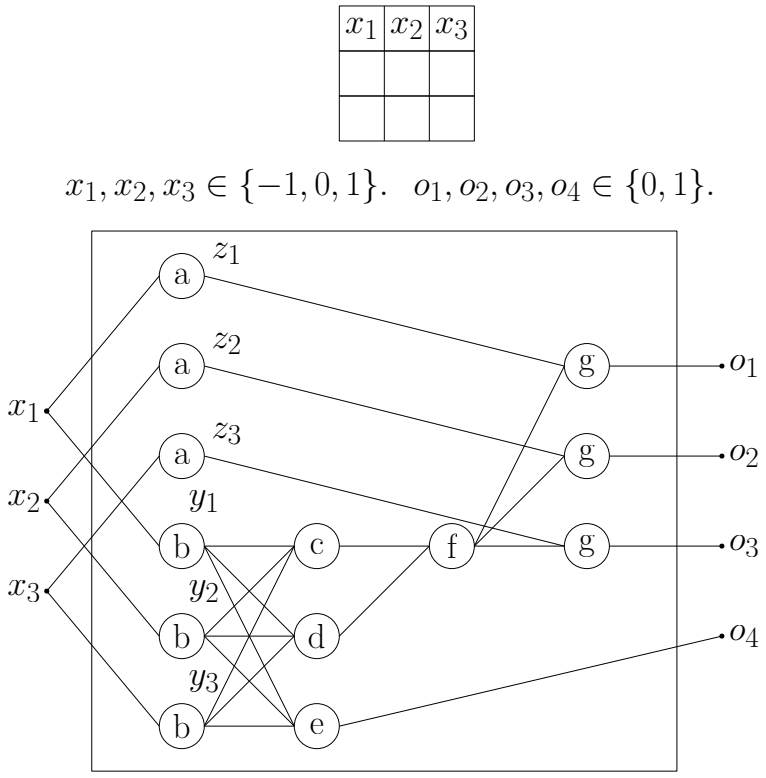
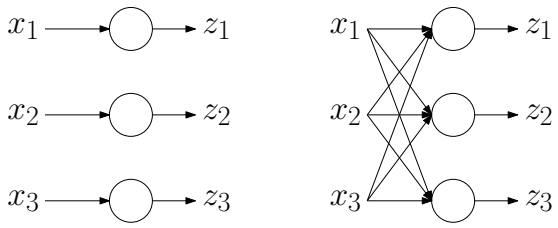


Figure 5: A network of McCulloch-Pitts Neurons could make decision for a step in a Tic-Tac-Toe game – Should a symbol be put on the cell corresponding to x_1, x_2 or x_3? In this network, there are seven different types of neurons. Some of them are single-input-single-output neurons (a-type and b-type). Some of them are two-input-single-output neurons (f-type and g-type) and some of them are three-input-single-output neurons (c-type, d-type and e-type). Here, if $o_4 = 1$, it means *game over*. Note that there are eight lines to be diagnosed – three rows, three columns and two diagonals. Decision on the next move in a tic-tac-toe game can be solved by a network consisting of eight of this network of McCulloch-Pitts Neurons.



$$z_1 = h(-x_1 + 0x_2 + 0x_3 - 0.5)$$
$$z_2 = h(0x_1 - x_2 + 0x_3 - 0.5)$$
$$z_3 = h(0x_1 + 0x_2 - x_3 - 0.5)$$

Figure 6: Implementation of an one-input-one-output neuron by three-input-one-output neuron. For the redundant weights, we simply set them to be zeros.

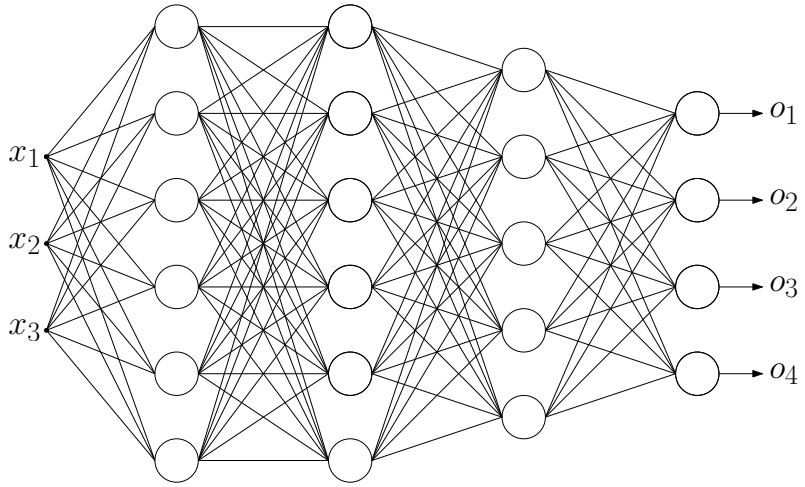


Figure 7: A 3-input-4-output multilayered network of all N-input-one-output neurons. Note that this network is also called a computational model. With proper design on the values for the weights and biases, this network is able to (but not limited to) replicate the functionalities of the network as shown in Figure 5.

Table 1: Complexity of some McCulloch-Pitts neuronal networks. The last model denoted as $(N_0 - \cdots - N_L)$ is a N_0-input-N_L-output network.

Model	**No. of Neurons**	**No. of Parameters**
AND	1	3
OR	1	3
NAND	1	3
NOR	1	3
XOR	3	9
Figure 5	13	36
Figure 7	21	125
$(N_0 - \cdots - N_L)$	$\sum_{k=1}^{L} N_k$	$\sum_{k=1}^{L} N_k \left(N_{k-1} + 1\right)$

1.7.1 Number of Processing Nodes

For a digital computer, a processing node refers to a logic gate which is a two-input-one-output system. For a McCulloch-Pitts neuron, it is an N-input-one-output processing node. In terms of the number of processing nodes, a McCulloch-Pitts neuronal network could be structural simpler than a digital computer.

An obvious example is on the number of inputs. A logic gate can only accept two inputs, while a McCulloch-Pitts neuron can accept more than two inputs. For the logical operation with three inputs and its output '1' if and only if all three inputs are '1', two AND logic gates are needed for this operation. Using McCulloch-Pitts neuron, we need only one. The network complexity could be reduced.

1.7.2 Beyond Digital Computations

Moreover, McCulloch-Pitts neuron accepts scalar inputs instead of binary. This neuron can be designed to solve problems with scalar inputs. Therefore, a network of McCulloch-Pitts neurons can be designed to solve 2-class classification problems – object recognition problems in which only two classes of objects are to be recognized. Along this line of thought, multiple networks of McCulloch-Pitts neurons can thus be applied to general object recognition problems with multiple classes of objects to be recognized. Furthermore, the model of McCulloch-Pitts neuron was applied in signal processing [3].

1.8 M-P Network as a Computational Model

It is no doubt that a network of McCulloch-Pitts neurons is essentially a computational model. As long as all the neuronal models have been defined, the operations of the network are defined accordingly. Each neuron simply performs a computation and gives results. The computational models developed along

Table 2: Interpretations of the variables and parameters in a M-P neuron.

Input $x_i = 1$	Electric pulse stream of a fixed firing rate r.
Input $x_i = 0$	No pulse stream received.
Output $f(\cdot) = 1$	Electric pulse stream of a fixed firing rate r.
Output $f(\cdot) = 0$	No pulse stream generated.
$w_i > 0$	Excitatory synapse.
$w_i = 0$	No connection.
$w_i < 0$	Inhibitory synapse.

this line are called *Perceptrons* which are developed and advocated by Frank Rosenblatt in the 1950s to 1960s [4, 5, 6].

In the example delineated in Figure 5, all parameters in the network are pre-defined by me. One question is then aroused. *What if the parameters are not given, is it possible to develop a learning algorithm for this model to get these parameters?* The answer is clearly YES. The learning rule associated with *Perceptrons* were later named as *Perceptron learning rule* in [2].

1.9 Interpretation of an M-P Neuron

One question regarding the M-P neuron is on the interpretations of the input and the output. If $x_1 = 1$, the McCulloch-Pitts receives a electric pulse stream of a fixed firing rate, say r. If the output of a McCulloch-Pitts neuron is one, the neuron generates a stream of electric pulses with firing rate r to the subsequent neurons. Table 2 summaries the physical meanings of the parameters in an M-P neuron.

1.10 Learning Classification

Figure 8 shows the use of a single 2-input-1-output McCulloch-Pitts neuron for data classification. The data in Group I is indicated by a circle and the data in Group II is indicated by a square. The main problem is to find the parameters w_1, w_2 and b for the decision boundary $u(x_1, x_2)$.

1.10.1 Step 1: Indexing, labeling and assessment

To solve this classification problem, the first step is to assign indices and labels for the data.

Indexing and labeling. Suppose the total number of data is N. We assign each data a unique index. For the k^{th} data, $\mathbf{x}_k = (x_{k1}, x_{k2})$ and d_k be respectively the coordinate and label of the k^{th} data. Its label d_k is defined as

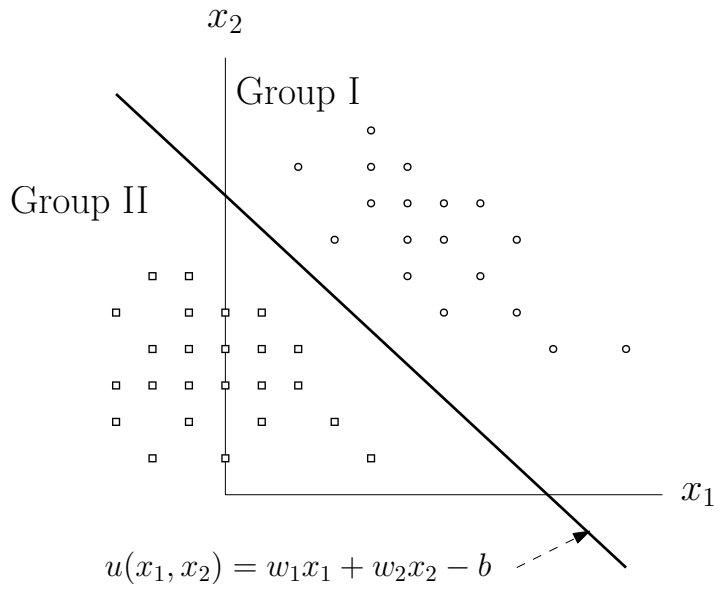


Figure 8: Use of a single 2-input-1-output McCulloch-Pitts neuron for data classification. The data in Group I is indicated by a circle and the data in Group II is indicated by a square.

follows[2] :

$$d_k = \begin{cases} 1 & \text{if } \mathbf{x}_k \text{ is in } \textit{Group I}, \\ 0 & \text{if } \mathbf{x}_k \text{ is in } \textit{Group II}. \end{cases} \tag{17}$$

Assessment. To assess how good a neuron with parameters w_1, w_2 and b can perform, we need to define a reasonable assessment measure. One can define the measure as the total prediction errors[3].

$$E(w_1, w_2, b) = \sum_{k=1}^{N} |d_k - f(\mathbf{x}_k)|, \tag{18}$$

where $f(\mathbf{x}_k)$ is the prediction of the neuron on the group to which the data $\mathbf{x}_k$ belongs. If the prediction is identical to the actual label, $|d_k - f(\mathbf{x}_k)| = 0$. Otherwise, $|d_k - f(\mathbf{x}_k)| = 1$. The values of $|d_k - f(\mathbf{x}_k)| = 1$ are depicted in Table 3 for clarification. In other words, $d_k - f(\mathbf{x}_k)| = 0$ *if only if the prediction is correct.* So, $E(w_1, w_2, b)$ is the total prediction errors of a neuron with model parameters w_1, w_2 and b. If $E(w_1, w_2, b) = 0$, the neuron with parameters w_1, w_2 and b is an optimal model.

[2]Note that this labelling is arbitrary. One can define $d_k = 0$ if $\mathbf{x}_k$ is in *Group I* and $d_k = 1$ if $\mathbf{x}_k$ is in *Group II*.

[3]One should be noted that the total prediction errors $E(w_1, w_2, b)$ is a non-differentiable function. Obtaining a learning rule which minimizes this function is not easy.

Table 3: Values of $|d_k - f(\mathbf{x}_k)|$.

d_k	$f(\mathbf{x}_k)$	$\lvert d_k - f(\mathbf{x}_k)\rvert$
0	0	0
0	1	1
1	0	1
1	1	0

Next, in search of (w_1, w_2, b)**.** With the above labelling, the second step is to develop a method to find the values w_1, w_2 and b their corresponding *total prediction errors* $E(w_1, w_2, b)$ is a minimum. Here, two methods are introduced, namely brute-force search and Perceptron learning.

1.10.2 Step 2: Brute-force search

Its key idea is to search all possible combinations of (w_1, w_2, b). For instance,

$$\begin{aligned} w_1 &= -5, -4.99, -4.98, \cdots, 4.98, 4.99, 5. \\ w_2 &= -5, -4.99, -4.98, \cdots, 4.98, 4.99, 5. \\ b &= -5, -4.99, -4.98, \cdots, 4.98, 4.99, 5. \end{aligned}$$

In such case, the total number of combinations of (w_1, w_2, w_3) is 1001^3. It is more than 10^9 combinations. For each (w_1, w_2, b), we feed in the data one by one to the inputs of the neuron and then calculate the neuronal output. Finally, the performance of this neuron $E(w_1, w_2, b)$ is calculated. Repeating the process for all 1001^3 combinations, we will have 1001^3 values of $E(w_1, w_2, b)$. In the end, those models with zero prediction error are the optimal models.

It is clear that *brute-force search* is not an efficient method to obtain an optimal model. For the number of parameters is larger, this method is infeasible. However, for some learning problems, this method is still a key for the search of model parameters.

1.10.3 Step 2: Perceptron learning

Long in the history, developing an efficient learning rule for a network of M-P neurons has been a challenging problem. Perceptron learning is one learning developed by Frank Rosenblatt in the 1950s [4, 5, 6]. For Perceptron learning, there are two modes of learning : batch mode and online mode.

Batch mode. For the batch mode, the M-P neuron predicts the labels for all N data. That is to say, the M-P neuron calculates $f(\mathbf{x}_k)$ for $k = 1, \cdots, N$. Then, these predictions are then compared with the actual labels to get $(d_k - f(\mathbf{x}_k))$ for $k = 1, \cdots, N$. Subseqently, the parameters w_1, w_2 and b are updated based

on the following equations.

$$w_1(t+1) = w_1(t) + \mu \sum_{k=1}^{N} (d_k - f(\mathbf{x}_k)) x_{k1}, \tag{19}$$

$$w_2(t+1) = w_2(t) + \mu \sum_{k=1}^{N} (d_k - f(\mathbf{x}_k)) x_{k2}, \tag{20}$$

$$b(t+1) = b(t) - \mu \sum_{k=1}^{N} (d_k - f(\mathbf{x}_k)), \tag{21}$$

where $w_1(0)$, $w_2(0)$ and $b(0)$ are arbitrary numbers. In (19), (20) and (21), the factor μ is called the learning step size which value is usually set to be a small number, say $\mu = 0.001$.

Online mode. In contrast to the batch mode learning, the update of w_1, w_2 and b is conducted one data at a time. Once a data $(\mathbf{x}_t, d_t)$ is *randomly selected* from the dataset, the M-P neuron calculates the prediction $f(\mathbf{x}_t)$ and then $(d_t - f(\mathbf{x}_t))$. Subsequently, the parameters w_1, w_2 and b are updated based on the following equations.

$$w_1(t+1) = w_1(t) + \mu_t (d_t - f(\mathbf{x}_t)) x_{t1}, \tag{22}$$

$$w_2(t+1) = w_2(t) + \mu_t (d_t - f(\mathbf{x}_t)) x_{t2}, \tag{23}$$

$$b(t+1) = b(t) - \mu_t (d_t - f(\mathbf{x}_t)), \tag{24}$$

where μ_t is a small number corresponding for the learning step size at time t, say $\mu_t = 0.01/t$. Besides, $w_1(0)$, $w_2(0)$ and $b(0)$ are arbitrary numbers. It can be shown that with proper setting[4] on μ_t, the online learning rule as stated in (22), (23) and (24) is able to get (precisely, *converge to*) an optimal model for the classification problem.

1.11 Illustrative Examples

Either for the batch mode learning as stated in (19), (20) and (21) or the online mode learning as stated in (22), (23) and (24), one should see that the update of the model parameters is relied on those data whose predictions are incorrect.

1.11.1 Separable data

To illustrate the behavior of the Perceptron learning rule, a set of two groups of data are randomly generated and shown in Figure 9. In this dataset, 100 data are belongs to *Group I* and 100 data are belongs to *Group II*. It is clear from Figure 9 that these two groups of data are separable.

[4]The conditions are that $\sum_{t=1}^{\infty} \mu_t = \infty$ and $\sum_{t=1}^{\infty} \mu_t^2 < \infty$.

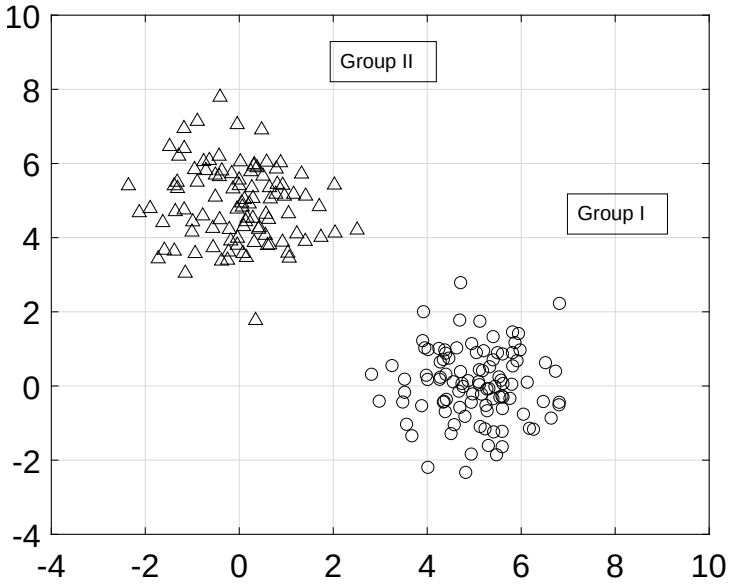


Figure 9: Two groups of data which are separable.

1.11.2 Settings

To determine the model parameters w_1, w_2 and b, the online mode Perceptron learning rule as stated in (22), (23) and (24) is applied with $\mu_t = 0.005$ for all t and the maximum of iteration is set to be 2000. Two initial conditions are simulated : (a) $w_1(0) = w_2(0) = b = 1$ and (b) $w_1(0) = w_2(0) = b = 0$.

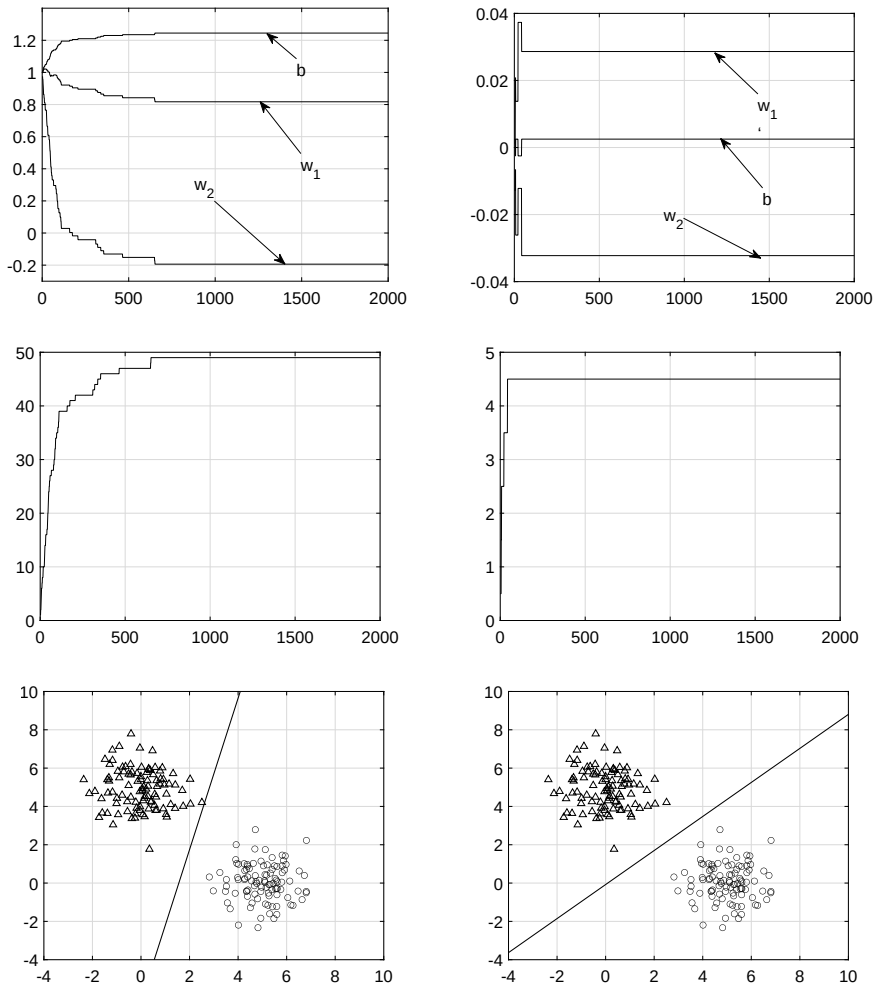
1.11.3 Results

Figure 10(**Top**) shows the changes of the parameters w_1, w_2 and b obtained by the online Perceptron learning rule over time $t = 1, \cdots, 2000$. Figure 10(**Middle**) shows the changes of the prediction errors $\sum_{k=1}^{t} |d_k - f(\mathbf{x}_k)|$ over time from $k = 1$ to $k = t$. The decision boundaries obtained are shown in Figure 10(**Bottom**).

It should be noted that the results shown in Figure 10 could be slightly difference if the same experiment is repeated. It is because of the online learning. In each step, the data to be selected is random. Therefore, sequence of data being selected for update in an experiment is clearly different from the sequence of data being selected in another experiment. The results shown in Figure 10(a) or Figure 10(b) are corresponding to one experiment, not for all.

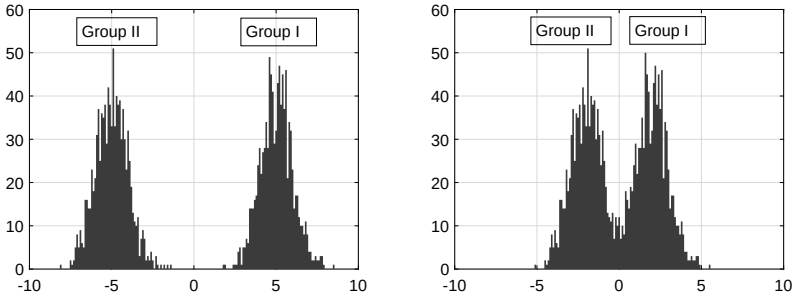
1.11.4 Comments

Applying Perceptron learning rule for a single M-P neuron, one needs to set the values for the initial conditions of w_1, w_2 and b. Besides, the learning rate μ and the maximum number of iterations have to be set. Different initial conditions



(a) $w_1(0) = w_2(0) = b(0) = 1$. (b) $w_1(0) = w_2(0) = b(0) = 1$.

Figure 10: Changes of the parameters w_1, w_2 and b over time for the dataset as shown in Figure 9. (a) With the initial condition $(w_1(0), w_2(0), b(0)) = (1, 1, 1)$, the parameters converge to $(0.8019, -0.2029, 1.2500)$ after $t \geq 750$. (b) With the initial condition $(w_1(0), w_2(0), b(0)) = (0, 0, 0)$, the parameters converge to $(0.0286, -0.0322, 0.0025)$ after $t \geq 100$. **Top:** Changes of parameters. **Middle:** Prediction errors. **Bottom:** Decision boundary.



(a) Separable. (b) Non-Separable.

Figure 11: Separable and non-separable data. For separable dataset, it is able to find an M-P neuron its total prediction errors is zero. For non-separable dataset, the minimum total prediction errors must be non-zero.

of w_1, w_2 and b might give different values of the convergent w_1, w_2 and b, i.e. different models. For the learning rate μ and the maximum number of iteration, the smaller the value of μ will lead to larger number of iterations. The settings of all these factors are basically determined by trial-and-error, i.e. by the experience of the developer.

1.12 Pitfall of a Network of M-P Neurons

A pitfall of the network of McCulloch-Pitts neurons is clearly on the development of a learning rule for multilayered M-P neuronal networks. For the case of single M-P neuron, the learning rule as stated in (22), (23) and (24) is able to let the neuron to attain an optimal for two-class linear separable classification problems. For a classification problem which is not linear separable, learning rule is difficult to be developed as the neuronal output is a step function.

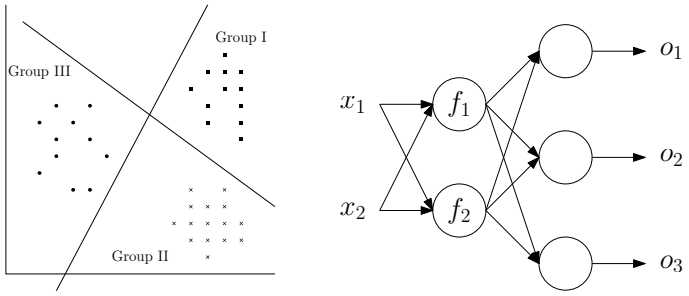
Figure 11 shows two examples. For either example, a good 1-input-1-output M-P neuron can be defined as follows :

$$f(x) = h(x), \text{ i.e. } w = 1,\ b = 0.$$

For the dataset as shown in Figure 11a, this model gives perfect predictions to all data, i.e. $E(1, 0) = 0$. However, for the dataset as shown in Figure 11b, $E(1, 0) > 0$.

1.13 Network of M-P Neurons for 3-Class Data

Applying the network of M-P neurons, it could be difficult to get a learning rule for a 3-class data classification problem. Figure 12 shows the distributions of the three classes of data and the Perceptron model which is able to solve this classification problem. The M-P neurons in the first layer perform the two decisions as indicated in Figure 12(a). Once the decision boundaries have been



(a) Three classes. (b) Perceptron model.

Figure 12: Three-Class classification problem. (a) Geometrical illustration of the distributions of the three classes of data. (b) The Perceptron model which can solve this classification problem.

obtained, the neurons at the output layer simply perform the logical operations depicted below.

$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	o_1	o_2	o_3	Group
0	0	0	0	0	–
0	1	1	0	0	I
1	0	0	0	1	III
1	1	0	1	0	II

It is clear that the Perceptron model as shown in Figure 12(b) can be designed to solve the 3-class classification problem. However, the learning rule for the update of the model parameters is not easily defined. Nevertheless, learning rule for the update of the model parameters in a multilayered M-P neuronal network is even difficult.

2 Sigmoidal Neuronal Networks

For a multilayered network of McCulloch-Pitts neurons, as shown in Figure 7, developing a learning rule for this network is difficult as the neuronal function is non-differentiable. Techniques from functional approximation and parametric estimation are not applicable, as those techniques require the (transfer function) model is differentiable.

In this regard, Paul Werbos in 1974 [7] suggested replacing the McCulloch-Pitts neuron by a differentiable function its shape is similar to an M-P neuron. Later, Rumelhart, Hinton and Williams [8] independently in 1986 suggested the same replacement. While Paul Werbos did not specify which differentiable function for a neuron model, Rumelhart, Hinton and Williams specifically introduced the sigmoid function as the neuron model. By that, the output of a

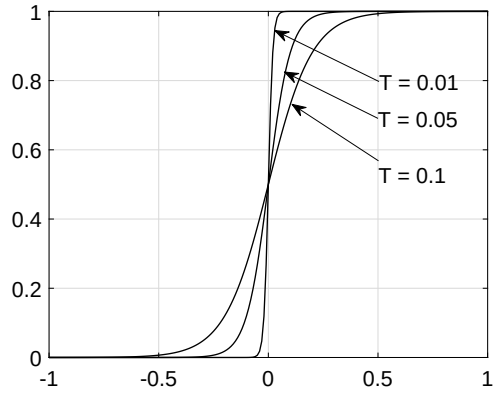


Figure 13: Sigmoid function could be considered as a relaxation of the step function in the McCulloch-Pitts model. The plots show the output versus the value of u. If $T \to 0$, the output is identical to a step function as in the McCulloch-Pitts model.

neuron is given by

$$f(x_1, x_2) = \frac{1}{1 + \exp\left(u(x_1, x_2)/T\right)} \tag{25}$$

$$= \frac{1}{1 + \exp\left((w_1 x_1 + w_2 x_2 - b)/T\right)} \tag{26}$$

where $u(x_1, x_2) = w_1 x_1 + w_2 x_2 - b$ as usual and the factor T is called the temperature. Figure 13 shows the plots of the output of a neuron against the input u for $T = 0.01$, $T = 0.05$ and $T = 0.1$.

2.1 Sigmoid Neuron

For a neuron with n inputs $\mathbf{x} = (x_1, \cdots, x_n)^T$, the output of a neuron is modeled as follows :

$$f(\mathbf{x}) = \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_i - b\right)}. \tag{27}$$

It should be noted that the temperature factor T is absorbed (redundant) in the parameters, i.e. $w_i \leftarrow w_i/T$ and $b \leftarrow b/T$.

2.2 Interpretation of a Sigmoid Neuron

Similar to that of a McCulloch-Pitts neuron, the physical meaning of the inputs, the outputs and the weights can be interpreted. In contrast to the M-P neuron, the value of an input to a sigmoid neuron is the *firing rate* of the impulse stream received from the input neuron. The sign of a weight w_i indicates if the connection is excitatory or inhibitory. The output of a neuron is the *firing rate* of the impulse steam to be generated. This interpretation is usually called the *rate coding* system.

2.3 Multilayered Perceptron (MLP)

Therefore, the multilayered neuronal network with this sigmoid neuron is then called a multilayered Percertron (MLP) or back-propagation network (BPN). From a mathematical function point of view, MLP is just a multiple-input-multiple-output function which can be denoted as $\mathbf{f}(\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ is the input and $\mathbf{w}$ is the vector of the function parameters.

2.4 Backpropagation (BP) Learning

For a sigmoid multilayered Percetron (MLP), the learning algorithm as proposed by Rumelhart *et al* is called backpropagation (BP). The learning algorithm is basically a gradient descent algorithm in search of the model parameters $\mathbf{w}$) in which its prediction errors $E(\mathbf{w})$ is a minimum. Here, the parameters of an MLP is denoted as a vector $\mathbf{w}$. Thus, the online learning for an MLP is given as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \mu_t \frac{\partial E(\mathbf{w}(t))}{\partial \mathbf{w}}, \tag{28}$$

where $E(\mathbf{w}(t))$ is the total prediction errors as follows :

$$E(\mathbf{w}) = \sum_{k=1}^{N} (d_k - f(\mathbf{x}_k, \mathbf{w}))^2. \tag{29}$$

Here, one should be noted that the total prediction errors as stated in (29) is different from that defined in (18).

2-Input-1-Output Sigmoid Neuron. For a two-input-one-output neuron, $\mathbf{w} = (w_1, w_2, b)$, the learning algorithm as stated in (28) can be stated as follows :

$$\begin{aligned} w_1(t+1) &= w_1(t) + \mu_t e(t) f'(\mathbf{x}_t, \mathbf{w}(t)) x_{t1} & (30)\\ w_2(t+1) &= w_2(t) + \mu_t e(t) f'(\mathbf{x}_t, \mathbf{w}(t)) x_{t2} & (31)\\ b(t+1) &= b(t) - \mu_t e(t) f'(\mathbf{x}_t, \mathbf{w}(t)), & (32) \end{aligned}$$

where

$$\begin{aligned} e(t) &= d_t - f(\mathbf{x}_t, \mathbf{w}(t)\\ f'(\mathbf{x}_t, \mathbf{w}(t)) &= f(\mathbf{x}_t, \mathbf{w}(t))(1 - f(\mathbf{x}_t, \mathbf{w}(t))). \end{aligned}$$

Again, the factor μ_t is the learning step at the time t. If μ_t satisfies a proper condition[5], it can be shown that BP learning can obtain a model (w_1, w_2, b) such that its $E(\mathbf{w})$ is a minimum.

[5]The conditions are that $\sum_{t=1}^{\infty} \mu_t = \infty$ and $\sum_{t=1}^{\infty} \mu_t^2 < \infty$.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] M. Minsky and S. Papert, *Perceptrons : An introduction to computational geometry*. MIT Press, 1969.

[3] B. Widrow, "Generalization and information storage in networks of Adaline neurons," in *Self-Organizing Systems*. Spartan Books, 1962, pp. 435–461.

[4] F. Rosenblatt, "The Perceptron: a probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[5] ——, "Perceptron simulation experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960.

[6] ——, *Principles of Neurodynamics: Perceptions and the theory of brain mechanisms*. Spartan, 1962.

[7] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *PhD Dissertation, Harvard University*, 1974.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.