Perceptron model

September 21, 2026

McCulloch-Pitts model and Perceptron

some historical account

Warren McCulloch and Walter Pitts in their 1943 paper, "A Logical Calculus of Ideas Immanent in Nervous Activity," proposed that neurons with a binary threshold activation function were analogous to first-order logic sentences. The McCulloch-Pitts neuron used binary inputs (1 for true, 0 for false) and produced a binary output based on a threshold value, typically 1.

A key issue with the McCulloch-Pitts neuron was its lack of a learning mechanism. In 1949, Donald Hebb introduced a synaptic learning rule in his book, The Organization of Behavior, known as Hebb's rule: "When an axon of cell A repeatedly or persistently helps fire cell B, some growth or metabolic change occurs in one or both cells, increasing A's efficiency in firing B." Hebb suggested that simultaneous firing of two neurons strengthens their connection, a fundamental process for learning and memory.

Frank Rosenblatt, building on the McCulloch-Pitts neuron and Hebb's findings, developed the first perceptron, capable of learning by adjusting its weights incrementally. In his 1962 book, Principles of Neurodynamics, Rosenblatt claimed: "Give an elementary α perceptron, a stimulus world $\mathcal{W}$, and any classification $C(\mathcal{W})$ with a solution; If all stimuli in $\mathcal{W}$ occur in any sequence and reoccur in finite time, an error correction procedure will always find a solution to $C(\mathcal{W})$ in finite time."

Rosenblatt's claims created a divide between perceptron research and traditional symbol manipulation projects by Marvin Minsky. In 1969, Minsky and Seymour Papert co-authored Perceptrons: An Introduction to Computational Geometry, highlighting the perceptron's limitations, notably its inability to solve XOR and NXOR functions. They argued that perceptron research was doomed due to these limitations, leading to a decline in research until the 1980s.

Perceptron Decision Surface

The simplest network that implements interesting input-output function is the Perceptron. It has a single layer of weights connecting the inputs and output. Formally, the perceptron is defined by

$$y = \text{sign}\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \tag{1}$$

In vector form we have $y = \text{sign}(\mathbf{w}^\top \mathbf{x} - \theta)$, where θ is the threshold parameter. Often, we ignore the threshold in perceptron analysis, treating it as another synaptic weight with a constant input of -1. This is because $\mathbf{w}^\top \mathbf{x} - \theta = [\mathbf{w}, \theta]^\top [\mathbf{x}, -1]$.

A perceptron performs a dichotomy, that is, a function from $\mathbb{R}^N$ to $\{-1, 1\}$ (or $\{0, 1\}$). The perceptron's action is visualized geometrically since $\mathbf{w}$ and $\mathbf{x}$ both have dimensionality N. The surface dividing points into two classes is a hyperplane defined by $\mathbf{w}^\top \mathbf{x} = 0$, passing through the origin and perpendicular to $\mathbf{w}$. With a non-zero threshold, the plane is $\mathbf{w}^\top \mathbf{x} - \theta = 0$, perpendicular to $\mathbf{w}$ and separated from the origin by $\theta/\|\mathbf{w}\|$. The plane is often referred to as the decision boundary. The perceptron realizes linearly separable dichotomies, a small subset of all possible dichotomies.

The capacity of a perceptron

The capacity of a network is the number of functions (dichotomies) it can implement on a fixed set of inputs $\{\mathbf{x}^1, \ldots, \mathbf{x}^P\}$. For a perceptron, the architecture is defined by $y = \text{sign}(\mathbf{w}^\top \mathbf{x})$ where $\mathbf{w}$ is an N-dimensional vector. The set of possible dichotomies over $\{\mathbf{x}^1, \ldots, \mathbf{x}^P\}$ is $\{\text{sign}(\mathbf{w}^\top \mathbf{x}^1), \ldots, \text{sign}(\mathbf{w}^\top \mathbf{x}^P) \mid \mathbf{w} \in \mathbb{R}^N\}$. The size of this set is the network's capacity. Generally, capacity depends on the specific inputs, but for a perceptron it is independent of the input vectors' identity if they are in general position.

The general position means $\{\mathbf{x}^1, \ldots, \mathbf{x}^P\}$ does not have a subset of size N or less that is linearly dependent. If the inputs are in general position, the capacity depends only on the input dimension N and the number of inputs P, denoted as $C(P, N)$. Clearly, $C(P, N) \leq 2^P$ since the total number of dichotomies over P inputs is 2^P. Cover's Function Counting Theorem (1966) provides the exact value of $C(P, N)$:

$$C(P, N) = 2 \sum_{k=0}^{N-1} \binom{P-1}{k} \tag{2}$$

The binomial coefficient is defined as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

for $n \geq k$, otherwise $\binom{n}{k} = 0$.

If $P \leq N$, the vectors are generally in general position, and Cover's Theorem applies. For $P > N$, the vectors are linearly dependent, but Cover's Theorem still holds if they are in general position, which is typical for generic input vectors. Thus, Cover's theorem is broadly applicable.

Some consequences of Cover's theorem:

1. For $P \leq N$, the above sum is limited by $P - 1$, so it can be rewritten as

$$C(P, N) = 2 \sum_{k=0}^{P-1} \binom{P-1}{k} = 2(1+1)^{P-1} = 2^P$$

by using the familiar binomial expansion. In other words, all possible dichotomies can be realized.

2. for $P = 2N$,

$$C(2N, N) = 2 \sum_{k=0}^{N-1} \binom{2N-1}{k} = 2\frac{1}{2}(1+1)^{2N-1} = 2^{P-1}$$

This occurs because the expression for C includes precisely half of the complete binomial expansion of $(1+1)^{2N-1}$. Given that this expansion is symmetric (for instance, the initial few (odd) binomial expansions are (1, 1), (1, 3, 3, 1), and (1, 5, 10, 10, 5, 1)), we deduce that exactly half of all potential dichotomies can be achieved in this scenario.

3. For $P \gg N$, $C(P, N) \sim AP^N$ for some $A > 0$.

4. Another way to view $C(P, N)$ is to let P and N approach ∞ while keeping $N/P = \alpha$. As $N \to \infty$, $C(P, N)$ vs. N/P forms a step function. The step occurs at $N/P = 2$, where $C(2N, N) = 0.5$. Below $N/P = 2$, almost all dichotomies are possible; above it, almost none are possible.

Proof of Cover's Theorem

We assume throughout that the patterns are generic, in the sense that no N or fewer of them are linearly dependent.[1]

Adding one pattern

The natural move is to build C up one pattern at a time. Suppose we already know the $C(P, N)$ realizable labellings of $\mathbf{x}^1, \ldots, \mathbf{x}^P$ and now present one more pattern, $\mathbf{x}^{P+1}$.

[1] This is the condition called *general position*. It excludes accidental degeneracies — three patterns lying in a common plane when $N = 3$, say — and holds with probability one for patterns drawn from any smooth distribution.

Any solution $\mathbf{w}$ for a labelling vector $\mathbf{y}_0 = (y_0^1, \ldots, y_0^P) \in \{-1, 1\}^P$ must satisfy the P strict inequalities

$$y_0^\mu \, \mathbf{w}^\top \mathbf{x}^\mu > 0, \qquad \mu = 1, \ldots, P. \tag{3}$$

Such a solution $\mathbf{w}$ automatically assigns some label to the new pattern as well, namely $\text{sign}(\mathbf{w}^\top \mathbf{x}^{P+1})$. So each realizable $\mathbf{y}_0$ gives us at least one realizable labelling of the enlarged set. The only way to do better is if $\mathbf{y}_0$ can be realized *twice over*, once with the new pattern on each side of the plane. Writing D for the number of labellings that can,

$$C(P+1, N) = C(P, N) + D. \tag{4}$$

Everything now hinges on D.

When are both signs available? Suppose first that among all the planes realizing $\mathbf{y}_0$ there is one that happens to pass exactly through $\mathbf{x}^{P+1}$, so that $\mathbf{w}^\top \mathbf{x}^{P+1} = 0$. Tilt it slightly, $\mathbf{w} \to \mathbf{w} \pm \varepsilon \mathbf{x}^{P+1}$. The P inequalities (3) are strict, so a small enough tilt cannot spoil them, while the new pattern acquires $\mathbf{w}^\top \mathbf{x}^{P+1} = \pm\varepsilon \|\mathbf{x}^{P+1}\|^2$. Both signs are therefore available, and $\mathbf{y}_0$ contributes to D.

The converse also holds, and it is what makes D a definite number rather than a lower bound. Suppose both signs are available, realized by planes $\mathbf{w}_+$ and $\mathbf{w}_-$. Deform one into the other along the straight path $\mathbf{w}_t = (1-t)\mathbf{w}_- + t\mathbf{w}_+$. Because the constraints (3) are linear, each quantity $y_0^\mu \mathbf{w}_t^\top \mathbf{x}^\mu$ is a positively weighted average of two positive numbers and stays positive: every plane along the way still classifies the first P patterns correctly. But $\mathbf{w}_t^\top \mathbf{x}^{P+1}$ starts negative and ends positive, so somewhere in between it vanishes — a plane realizing $\mathbf{y}_0$ and containing $\mathbf{x}^{P+1}$.

So D counts exactly those labellings of the original P patterns that can be realized by a plane pinned to pass through $\mathbf{x}^{P+1}$.

Pinning costs one dimension. Requiring $\mathbf{w}^\top \mathbf{x}^{P+1} = 0$ confines the weight vector to an $(N-1)$-dimensional subspace. Choose axes so that $\mathbf{x}^{P+1}$ points along one coordinate direction; the plane then has only the remaining $N-1$ directions to work with. Equivalently, splitting $\mathbf{x}^\mu = \mathbf{x}_\parallel^\mu + \mathbf{x}_\perp^\mu$ relative to $\mathbf{x}^{P+1}$, the parallel components drop out of $\mathbf{w}^\top \mathbf{x}^\mu$ altogether, and we are counting the labellings of the P projected patterns $\mathbf{x}_\perp^\mu$ in $N-1$ dimensions.[2]

This is the same counting problem one dimension down, so $D = C(P, N-1)$ and

$$\boxed{C(P+1, N) = C(P, N) + C(P, N-1)} \tag{5}$$

A new pattern buys you the old count plus whatever the problem yields in one fewer dimension.

[2]The projected patterns are again generic: a dependence $\sum_\mu c_\mu \mathbf{x}_\perp^\mu = 0$ among at most $N-1$ of them would mean $\sum_\mu c_\mu \mathbf{x}^\mu = \lambda \, \mathbf{x}^{P+1}$, a dependence among at most N of the original patterns, hence trivial.

Solving the recursion

Two boundary conditions fix the solution. In one dimension the weight is a single number, $\mathrm{sign}(wx^\mu) = \mathrm{sign}(w)\,\mathrm{sign}(x^\mu)$, so flipping the sign of w is the only freedom available and $C(P,1) = 2$ for all P. With one pattern, either label is obviously attainable, so $C(1,N) = 2$ for all N. Both match (2).

Equation (5) together with these edges determines C everywhere, and (2) satisfies it: with $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ and $\binom{n}{k} = 0$ for $k < 0$,

$$C(P,N)+C(P,N-1) = 2\sum_{k=0}^{N-1}\binom{P-1}{k}+2\sum_{k=0}^{N-1}\binom{P-1}{k-1} = 2\sum_{k=0}^{N-1}\binom{P}{k} = C(P+1,N). \tag{6}$$

The structure of the answer is worth pausing on. Pascal's rule is what generates the binomial coefficients, and it enters through (5): adding a pattern is formally the same operation as adding a row to Pascal's triangle. The truncation of the sum at $k = N-1$ is the whole geometric content — it is the statement that only N directions are available.

The thermodynamic limit

The interesting regime is $P, N \to \infty$ at fixed load

$$\alpha = \frac{P}{N}, \tag{7}$$

where the useful quantity is the *fraction* of labellings that survive, $f = C(P,N)/2^P$. Read (2) probabilistically: $2^{-(P-1)}\binom{P-1}{k}$ is the probability of k heads in $P-1$ fair coin tosses, so

$$f(P,N) = \mathrm{Prob}\big[\,k \le N-1\,\big], \qquad k \sim \mathrm{Binomial}(P-1, \tfrac{1}{2}). \tag{8}$$

The distribution has mean $\frac{P}{2}$ and standard deviation $\frac{1}{2}\sqrt{P}$, so for large N

$$f \simeq \Phi\left(\frac{N - P/2}{\frac{1}{2}\sqrt{P}}\right) = \Phi\left(\sqrt{N}\,\frac{2-\alpha}{\sqrt{\alpha}}\right), \tag{9}$$

with Φ the cumulative Gaussian. The argument carries a factor $\sqrt{N}$, so as $N \to \infty$ the fraction becomes a step:

$$f \to \begin{cases} 1, & \alpha < 2 \\ \frac{1}{2}, & \alpha = 2 \\ 0, & \alpha > 2 \end{cases} \tag{10}$$

The perceptron has a sharp capacity at $\alpha_c = 2$: below two patterns per weight essentially every labelling is learnable, above it essentially none. At the critical load exactly half survive, which is the elementary identity $C(2N,N) = 2^{P-1}$ — the binomial expansion is symmetric and the sum (2) retains exactly half of it.

The transition is sharp only in the limit. From (9) the crossover has width

$$\Delta\alpha \sim N^{-1/2}, \tag{11}$$

which is why the $N = 5$ curve in the figure is visibly rounded while the $N = 65$ curve is already close to a step. This is the familiar finite-size rounding of a sharp transition, with N playing the role of system size, and it has the usual $\sqrt{N}$ form because the underlying count is a sum of independent binary contributions.

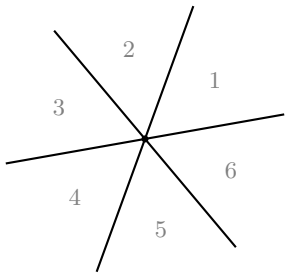
The same count seen from weight space

There is a second way to read the whole calculation which makes the binomial sum look inevitable. Each pattern defines a hyperplane $H_\mu = \{\mathbf{w} : \mathbf{w}^\top \mathbf{x}^\mu = 0\}$ in the space of weights. A label $\mathrm{sign}(\mathbf{w}^\top \mathbf{x}^\mu)$ can only change when $\mathbf{w}$ crosses H_μ, and a prescribed labelling confines $\mathbf{w}$ to a single wedge. Labellings and regions are the same objects:

$$C(P, N) = \#\{\text{regions cut out of } \mathbb{R}^N \text{ by } P \text{ generic hyperplanes through the origin}\}. \tag{12}$$

Adding a hyperplane, each existing region is either missed and survives, or is sliced in two. The increment is the number of regions the new hyperplane passes through, which is the number of pieces it is itself cut into by its intersections with the others. Those intersections are P hyperplanes through the origin of an $(N-1)$-dimensional space, giving $C(P, N-1)$ and reproducing (5).

In $\mathbb{R}^3$: three generic planes through the origin make $C(3,3) = 8$ regions. A fourth plane carries three lines through its origin, cutting it into $C(3,2) = 6$ sectors, so it slices six of the eight regions and leaves two intact, $C(4,3) = 8 + 6 = 14 < 2^4$. The two missing labellings are precisely the price of the linear dependence that four vectors in $\mathbb{R}^3$ cannot avoid.



Three lines through the origin of $\mathbb{R}^2$: $C(3,2) = 6$ sectors.

Perceptron learning Algorithm

Our goal is to train a perceptron to classify points. Given patterns $\{\mathbf{x}^1, \ldots, \mathbf{x}^P\}$ and labels $\{y_0^1, \ldots, y_0^P\}$, we seek a weight vector $\mathbf{w}$ such that $\mathrm{sign}(\mathbf{w}^\top \mathbf{x}^\mu) = y_0^\mu$ for all μ. We need an algorithm to find weights that separate the data.

The Perceptron learning algorithm updates incrementally, beginning with initial weights $\mathbf{w}_0$. At iteration n, the weights $\mathbf{w}_{n-1}$ and the example $(\mathbf{x}^n, y_0^n)$ are utilized. The examples are shown sequentially: $1, 2, \ldots, P, 1, 2, \ldots, P, \ldots$.

If $y_0^n(\mathbf{w}_{n-1}^\top \mathbf{x}^n) > 0$, the sample is classified correctly: assign $\mathbf{w}_n = \mathbf{w}_{n-1}$. If $y_0^n(\mathbf{w}_{n-1}^\top \mathbf{x}^n) < 0$, modify the weights: $\mathbf{w}_n = \mathbf{w}_{n-1} + \eta y_0^n \mathbf{x}^n$. The adjustment of the weights can be represented as

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \eta \left(y_0^n - y\right) \mathbf{x}^n$$

where

$$y = \text{sign}(\mathbf{w}_{n-1}^\top \mathbf{x}^n).$$

As previously stated, the training samples are presented sequentially and repeated from the start after the Pth sample is processed. Upon completing a full cycle from the first to the last sample, we count the errors made during that cycle (which corresponds to the number of updates to $\mathbf{w}$). If no errors are found, the algorithm stops, indicating that all training samples are correctly classified. Fortunately, the following theorem ensures that the algorithm is effective if the data is linearly separable:

Perceptron Convergence Theorem: If P examples are given that are linearly separable, then the Perceptron Learning Algorithm will stop after a finite number of iterations to a vector $\mathbf{w}$ of connections that will divide the examples without errors, i.e. $\text{sign}(\mathbf{w}^\top \mathbf{x}^\mu) = y_0^\mu$ for every μ. Note: It is true that the number of iterations is finite, but it is usually larger than P because each example needs to be processed more than once.

Proof sketch: We note $\mathbf{w}^*$ a weight vector that correctly separates the training examples: $\forall \mu,\ y_0^\mu = \text{sign}(\mathbf{w}^{*\top} \mathbf{x}^\mu)$. Let $\mathbf{w}_n$ be the student's weight vector in the n-th time step. Define the cosine similarity:

$$\cos(\theta_n) = \frac{\mathbf{w}_n^\top \mathbf{w}^*}{\|\mathbf{w}_n\| \, \|\mathbf{w}^*\|} \tag{13}$$

The fundamental concept of the proof is that, if the algorithm continues indefinitely, the RHS of this equation exceeds 1 for large n. This occurs because $\mathbf{w}_n$ undergoes a biased random walk during the learning process. Since the bias is directed towards $\mathbf{w}^*$, the numerator increases linearly with n. Conversely, because the change in $\mathbf{w}_n$ is biased away from the current weight vector $\mathbf{w}_{n-1}$, the term $\|\mathbf{w}_n\|$ in the denominator only increases as $\sqrt{n}$ and does not become large.

Convergence theorem

For linearly separable data, the algorithm makes at most $(D/\delta^*)^2$ *updates and then classifies every pattern correctly* (Rosenblatt 1962; Novikoff 1962). The

two quantities are properties of the data:

$$\delta^* = \min_\mu \frac{y^\mu \mathbf{w}^{*\top} \mathbf{x}^\mu}{\|\mathbf{w}^*\|} > 0, \qquad D = \max_\mu \|\mathbf{x}^\mu\|. \tag{14}$$

$\delta^\mu = y^\mu \mathbf{w}^{*\top} \mathbf{x}^\mu / \|\mathbf{w}^*\|$ is the distance of pattern μ from the plane $\mathbf{w}^*$, its *margin*; δ^* is the smallest margin and D the data scale.

Proof. Follow the angle θ_n between the weight vector after n updates and $\mathbf{w}^*$,

$$\cos\theta_n = \frac{\mathbf{w}_n^\top \mathbf{w}^*}{\|\mathbf{w}_n\| \, \|\mathbf{w}^*\|} \leq 1. \tag{15}$$

The overlap grows linearly. At the n-th update, on pattern μ,

$$\mathbf{w}_n^\top \mathbf{w}^* = \mathbf{w}_{n-1}^\top \mathbf{w}^* + \eta \, y^\mu \mathbf{w}^{*\top} \mathbf{x}^\mu \geq \mathbf{w}_{n-1}^\top \mathbf{w}^* + \eta \, \delta^* \|\mathbf{w}^*\|, \tag{16}$$

so $\mathbf{w}_n^\top \mathbf{w}^* \geq n\eta\delta^* \|\mathbf{w}^*\|$.

The norm grows only as $\sqrt{n}$.

$$\|\mathbf{w}_n\|^2 = \|\mathbf{w}_{n-1}\|^2 + \eta^2 \|\mathbf{x}^\mu\|^2 + 2\eta \, y^\mu \mathbf{w}_{n-1}^\top \mathbf{x}^\mu \leq \|\mathbf{w}_{n-1}\|^2 + \eta^2 D^2, \tag{17}$$

because the cross term is non-positive precisely when an update occurs. So $\|\mathbf{w}_n\|^2 \leq n\eta^2 D^2$.

Combining,

$$1 \geq \cos\theta_n \geq \frac{n\eta\delta^*}{\sqrt{n}\,\eta D} = \sqrt{n}\,\frac{\delta^*}{D} \quad \Longrightarrow \quad n \leq \left(\frac{D}{\delta^*}\right)^2. \tag{18}$$

The overlap with the solution accumulates coherently, one push per update, while the norm accumulates like the displacement of a random walk, because each update is non-positively aligned with the current weights. Coherent growth beats diffusive growth, and the ratio is capped. (An arbitrary $\mathbf{w}_0$ adds constant offsets to both bounds and does not change the conclusion.)