Simple Neural Nets : Perceptron

Ref : Fausett's Fundamentals of Neural Networks (1994).

1) What is a perceptron?

A (binary) perceptron is a linear classifier that maps an input vector $x \in \mathbb{R}^d$ to a class label $y \in \{-1, +1\}$ using:

- **Weighted sum (net input)**:

$$a = w^\top x + b$$

- **Hard-threshold activation**:

$$\hat{y} = \text{sign}(a) = \begin{cases} +1 & \text{if } a \geqslant 0 \\ -1 & \text{if } a < 0 \end{cases}$$

Here $w \in \mathbb{R}^d$ is the weight vector and $b \in \mathbb{R}$ is the bias. Many expositions (including Fausett) absorb the bias by augmenting x with a constant 1: $\tilde{x} = [1, x_1, \ldots, x_d]^\top$ and $\tilde{w} = [b, w_1, \ldots, w_d]^\top$, so $a = \tilde{w}^\top \tilde{x}$.

2) Learning rule (Perceptron update)

Given a training set $\{(x^{(i)}, t^{(i)})\}_{i=1}^N$ with targets $t^{(i)} \in \{-1, +1\}$, we iterate over examples and update **only when the current example is misclassified**:

- **Prediction**: $\hat{y}^{(i)} = \text{sign}(w^\top x^{(i)} + b)$
- **If** $\hat{y}^{(i)} \neq t^{(i)}$, update:

$$w \leftarrow w + \eta\, t^{(i)} x^{(i)}, \qquad b \leftarrow b + \eta\, t^{(i)}$$

Equivalently (augmented form): $\tilde{w} \leftarrow \tilde{w} + \eta\, t^{(i)} \tilde{x}^{(i)}$.

$\eta > 0$ is the learning rate (often $\eta = 1$). This rule nudges the decision boundary toward correctly classifying the offending point.

3) Worked Example A (fully detailed): Learning the OR function

We'll learn the Boolean **OR** with inputs in $\{0, 1\}^2$ and targets in $\{-1, +1\}$:

x_1	x_2	OR	t
0	0	0	−1
0	1	1	+1
1	0	1	+1
1	1	1	+1

@S S Roy
2025

Use the augmented form $\tilde{x} = [1, x_1, x_2]$, $\tilde{w} = [b, w_1, w_2]$.
Initialize $\tilde{w}^{(0)} = [0, 0, 0]$, learning rate $\eta = 1$.

Epoch 1 (go through all four patterns in order)

1. Example $(x = (0, 0), t = -1)$
   $\tilde{x} = [1, 0, 0]$.
   Net input $a = \tilde{w}^{(0)} \cdot \tilde{x} = 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 = 0$.
   Prediction $\hat{y} = \text{sign}(0) = +1$.
   Misclassified (want -1). **Update**:

$$\tilde{w}^{(1)} = \tilde{w}^{(0)} + \eta\, t\, \tilde{x} = [0, 0, 0] + 1 \cdot (-1) \cdot [1, 0, 0] = [-1, 0, 0].$$

2. Example $(x = (0, 1), t = +1)$
   $\tilde{x} = [1, 0, 1]$.
   $a = (-1) \cdot 1 + 0 \cdot 0 + 0 \cdot 1 = -1$.
   $\hat{y} = \text{sign}(-1) = -1 \rightarrow$ misclassified. **Update**:

$$\tilde{w}^{(2)} = [-1, 0, 0] + 1 \cdot (+1) \cdot [1, 0, 1] = [0, 0, 1].$$

3. Example $(x = (1, 0), t = +1)$
   $\tilde{x} = [1, 1, 0]$.
   $a = 0 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 = 0$.
   $\hat{y} = +1 \rightarrow$ correct. **No update**: $\tilde{w}^{(3)} = [0, 0, 1]$.
4. Example $(x = (1, 1), t = +1)$
   $\tilde{x} = [1, 1, 1]$.
   $a = 0 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 1$.
   $\hat{y} = +1 \rightarrow$ correct. **No update**: $\tilde{w}^{(4)} = [0, 0, 1]$.

Check convergence (another pass)

Run through the four again with $\tilde{w} = [0, 0, 1]$:

- $(0,0)$: $a = 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 = 0 \Rightarrow \hat{y} = +1$ (but **should be** -1). Misclassified.
  Update: $\tilde{w} \leftarrow [0,0,1] + (-1)[1,0,0] = [-1,0,1]$.

Now test all with $\tilde{w} = [-1,0,1]$:

- $(0,0)$: $a = (-1) \cdot 1 + 0 + 0 = -1 \Rightarrow \hat{y} = -1 =$ correct.
- $(0,1)$: $a = (-1) \cdot 1 + 0 + 1 \cdot 1 = 0 \Rightarrow \hat{y} = +1 =$ correct.
- $(1,0)$: $a = (-1) \cdot 1 + 0 \cdot 1 + 1 \cdot 0 = -1 \Rightarrow \hat{y} = -1$ (**should be +1**) → misclassified.
  Update on $(1,0)$: $\tilde{w} \leftarrow [-1,0,1] + (+1)[1,1,0] = [0,1,1]$.

@S S Roy
2025

Re-check all with $\tilde{w} = [0,1,1]$:

- $(0,0)$: $a = 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 = 0 \Rightarrow \hat{y} = +1$ (needs -1) → misclassified.
  Update: $[0,1,1] + (-1)[1,0,0] = [-1,1,1]$.

Final test with $\tilde{w} = [-1,1,1]$:

- $(0,0)$: $a = -1 + 0 + 0 = -1 \Rightarrow -1$ ✓
- $(0,1)$: $a = -1 + 0 + 1 = 0 \Rightarrow +1$ ✓
- $(1,0)$: $a = -1 + 1 + 0 = 0 \Rightarrow +1$ ✓
- $(1,1)$: $a = -1 + 1 + 1 = 1 \Rightarrow +1$ ✓

Converged with decision function $\hat{y} = \text{sign}(-1 + x_1 + x_2)$. Geometrically, this is the half-space above the line $x_1 + x_2 = 1$.

4) Worked Example B: A 2D, real-valued, linearly separable set

Training set (two classes):

- Positive ($t = +1$): $(2,1),\ (2,3),\ (3,2)$
- Negative ($t = -1$): $(0,-1),\ (-1,-2),\ (-2,-1)$

Augment with bias: $\tilde{x} = [1, x_1, x_2]$. Start $\tilde{w}^{(0)} = [0,0,0]$, $\eta = 1$.

Pass 1

1. $(2,1), t = +1$: $a = 0 \Rightarrow \hat{y} = +1$ → correct, no update.
2. $(2,3), t = +1$: $a = 0 \Rightarrow \hat{y} = +1$ → correct, no update.
3. $(3,2), t = +1$: $a = 0 \Rightarrow \hat{y} = +1$ → correct, no update.
4. $(0,-1), t = -1$: $a = 0 \Rightarrow \hat{y} = +1$ (wrong).
   Update: $\tilde{w} = [0,0,0] + (-1)[1,0,-1] = [-1,0,1]$.
5. $(-1,-2), t = -1$: $a = (-1) \cdot 1 + 0 \cdot (-1) + 1 \cdot (-2) = -3 \Rightarrow \hat{y} = -1$ ✓
6. $(-2,-1), t = -1$: $a = (-1) \cdot 1 + 0 \cdot (-2) + 1 \cdot (-1) = -2 \Rightarrow \hat{y} = -1$ ✓

Pass 2 (verify all): with $\tilde{w} = [-1, 0, 1]$

- $(2, 1)$: $a = -1 + 0 \cdot 2 + 1 \cdot 1 = 0 \Rightarrow +1$ ✓
- $(2, 3)$: $a = -1 + 0 \cdot 2 + 1 \cdot 3 = 2 \Rightarrow +1$ ✓
- $(3, 2)$: $a = -1 + 0 \cdot 3 + 1 \cdot 2 = 1 \Rightarrow +1$ ✓
- Negatives remain correctly classified as above.

Converged. Decision boundary: $-1 + 0 \cdot x_1 + 1 \cdot x_2 = 0$ i.e., $x_2 = 1$. Everything with $x_2 \geqslant 1$ is classified $+1$; else -1. (Satisfies the listed samples.)

5) When (and why) the perceptron fails: XOR **(In below you can see this)**

Perceptrons can only separate **linearly separable** data. The classic counterexample is XOR:

x_1	x_2	XOR	t
0	0	0	−1
0	1	1	+1
1	0	1	+1
1	1	0	−1

No single straight line can separate positives $(0, 1), (1, 0)$ from negatives $(0, 0), (1, 1)$. The perceptron learning rule will keep cycling among weight vectors without convergence (updates continue indefinitely or until you stop), which motivates multi-layer networks with nonlinear activations.

6) Perceptron Convergence Theorem (intuition)

If the training set is linearly separable, the perceptron algorithm **converges in a finite number of updates** to a solution that correctly classifies all training points. High-level intuition (omitting a full formal proof here, but included conceptually in standard texts like Fausett):

- Assume there exists a separating hyperplane with margin $\gamma > 0$ and a separator w^{*} with $\| w^{*} \| = 1$ such that $t^{(i)} (w^{*\top} x^{(i)}) \geqslant \gamma$ for all i.
- Each mistake update increases the projection of the current weight onto w^{*} by at least γ, while the weight norm grows at most with the square root of the number of mistakes.
- Combining upper and lower bounds implies a finite bound on total mistakes $M \leqslant (\frac{R}{\gamma})^2$, where $R = \max_i \| x^{(i)} \|$. Thus, convergence occurs after finitely many updates.

7) Multiclass classification with perceptrons

A common approach: **one-vs-rest (OvR)**.

- Train K separate perceptrons, one for each class k, using targets $t_k = +1$ for class k and -1 for all others.
- At inference, compute all scores $a_k = w_k^\top x + b_k$ and pick $\arg\max_k a_k$.

Each perceptron uses the same update rule shown earlier, applied to its own relabeled data. For linearly separable OvR problems, each binary task can converge.

8) Practical details & tips

- **Feature scaling** helps the algorithm move more steadily (big features can dominate the dot-product).
- **Learning rate** η: any positive value works; $\eta = 1$ is common since the perceptron uses a sign loss (no smooth gradient).
- **Shuffling** the data each epoch can avoid cyclic visiting patterns.
- **Stopping**: stop after an epoch with zero mistakes, or after a preset max-epochs (if data is nonseparable).

9) A final mini–exercise (with full calculations)

Try learning the **AND** function:

(x_1, x_2)	t
$(0, 0)$	-1
$(0, 1)$	-1
$(1, 0)$	-1
$(1, 1)$	$+1$

$$\tilde{x} = [1, x_1, x_2],\ \ \tilde{w}^{(0)} = [0, 0, 0],\ \ \eta = 1.$$

Epoch 1

1. $(0, 0), t = -1$: $a = 0 \Rightarrow \hat{y} = +1 \rightarrow$ wrong.
   Update: $[0, 0, 0] + (-1)[1, 0, 0] = [-1, 0, 0]$.
2. $(0, 1), t = -1$: $a = (-1) \cdot 1 + 0 + 0 = -1 \Rightarrow -1$ ✓
3. $(1, 0), t = -1$: $a = (-1) \cdot 1 + 0 + 0 = -1 \Rightarrow -1$ ✓
4. $(1, 1), t = +1$: $a = (-1) \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1 \Rightarrow -1 \rightarrow$ wrong.
   Update: $[-1, 0, 0] + (+1)[1, 1, 1] = [0, 1, 1]$.

Epoch 2 (check all) with $\tilde{w} = [0, 1, 1]$

- $(0, 0)$: $a = 0 \Rightarrow +1$ (needs -1) $\rightarrow$ wrong.
  Update: $[0, 1, 1] + (-1)[1, 0, 0] = [-1, 1, 1]$.
- $(0, 1)$: $a = -1 + 0 + 1 = 0 \Rightarrow +1$ (needs -1) $\rightarrow$ wrong.
  Update: $[-1, 1, 1] + (-1)[1, 0, 1] = [-2, 1, 0]$.

- $(1, 0)$: $a = -2 + 1 + 0 = -1 \Rightarrow -1$ ✓
- $(1, 1)$: $a = -2 + 1 + 0 = -1 \Rightarrow -1$ (needs $+1$) → wrong.
  Update: $[-2, 1, 0] + (+1)[1, 1, 1] = [-1, 2, 1]$.

Epoch 3 (verify) with $\tilde{w} = [-1, 2, 1]$

- $(0, 0)$: $a = -1 \Rightarrow -1$ ✓
- $(0, 1)$: $a = -1 + 0 + 1 = 0 \Rightarrow +1$ (needs -1) → wrong.
  Update: $[-1, 2, 1] + (-1)[1, 0, 1] = [-2, 2, 0]$.
- $(1, 0)$: $a = -2 + 2 + 0 = 0 \Rightarrow +1$ (needs -1) → wrong.
  Update: $[-2, 2, 0] + (-1)[1, 1, 0] = [-3, 1, 0]$.
- $(1, 1)$: $a = -3 + 1 + 0 = -2 \Rightarrow -1$ (needs $+1$) → wrong.
  Update: $[-3, 1, 0] + (+1)[1, 1, 1] = [-2, 2, 1]$.

Epoch 4 (check) with $\tilde{w} = [-2, 2, 1]$

- $(0, 0)$: $-2 \Rightarrow -1$ ✓
- $(0, 1)$: $-2 + 0 + 1 = -1 \Rightarrow -1$ ✓
- $(1, 0)$: $-2 + 2 + 0 = 0 \Rightarrow +1$ (needs -1) → wrong.
  Update: $[-2, 2, 1] + (-1)[1, 1, 0] = [-3, 1, 1]$.
- $(1, 1)$: $-3 + 1 + 1 = -1 \Rightarrow -1$ (needs $+1$) → wrong.
  Update: $[-3, 1, 1] + (+1)[1, 1, 1] = [-2, 2, 2]$.

Epoch 5 (final check) with $\tilde{w} = [-2, 2, 2]$

- $(0, 0)$: $-2 \Rightarrow -1$ ✓
- $(0, 1)$: $-2 + 0 + 2 = 0 \Rightarrow +1$ (needs -1) → wrong.
  Update: $[-2, 2, 2] + (-1)[1, 0, 1] = [-3, 2, 1]$.
- $(1, 0)$: $-3 + 2 + 0 = -1 \Rightarrow -1$ ✓
- $(1, 1)$: $-3 + 2 + 1 = 0 \Rightarrow +1$ ✓

Now all four are correct with $\tilde{w} = [-3, 2, 1]$, i.e.

$$\hat{y} = \text{sign}(-3 + 2x_1 + x_2),$$

which classifies only $(1, 1)$ as $+1$ and the others as -1, matching AND. (Different training orders can converge faster; perceptron solutions are not unique.)

this page intentionally left blank