**Challenge 7: Animated Perceptron Learning**

**1. Introduction**

This project demonstrates the learning process of a single-layer perceptron using a 2D linearly separable dataset (OR gate). The key objective was to visualize how the perceptron updates its weights using the perceptron learning rule and how the decision boundary evolves over time.

**2. Dataset**

The dataset chosen for this challenge is the **OR gate**, a classic example of a linearly separable problem in binary classification.

|  |  |  |
| --- | --- | --- |
| **x1** | **x2** | **Output (y = x1 OR x2)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Each input is a 2D vector, and the output is a binary label (0 or 1).

**3. Perceptron Learning Rule**

The perceptron updates its weights using the following rule:



Where:

A black background with white text

AI-generated content may be incorrect.

A step activation function is used:

A black background with white text

AI-generated content may be incorrect.

**4. Implementation Details**

* Programming language: Python
* Libraries used: numpy, matplotlib, matplotlib.animation, IPython.display
* Bias term was included by augmenting input vectors
* Initial weights were randomly initialized
* Visualization was performed using matplotlib's animation tools
* Each update to the weights changed the decision boundary (a line in 2D), which was plotted

**5. Visualization Output**

The animation showcases the perceptron's learning process over several weight update steps:

* **Red point**: Represents the only input with label 0 — the point [0, 0].
* **Blue points**: Represent inputs with label 1 — points [0, 1], [1, 0], and [1, 1].

**Decision Boundary Behavior:**

* The animation starts with a **randomly oriented decision boundary** (black line) that misclassifies points.
* With each step (weight update), the line **adjusts position and slope**, responding to misclassified examples.
* The line shifts in accordance with the **perceptron learning rule**, gradually improving classification accuracy.
* In the final frame, the boundary **clearly separates** the [0, 0] point from all others, correctly modeling the OR gate.

**Animation Features:**

* The animation runs **step-by-step**, with each frame representing a new decision boundary after a weight update.
* The number of steps corresponds to the updates made across 10 epochs of training.
* The final boundary is **stable**, indicating the model has converged.

This visualization effectively illustrates the dynamic nature of learning in a single-layer perceptron and clearly satisfies the challenge objective.

**6. Output Artifacts**

* ✅ **Animated GIF:** Perceptron\_Learning\_Animation.gif (can be played to review learning process)
* ✅ **Python Script:** challenge7\_perceptron\_animation.py (can be reused to generate the visualization)
* ✅ **Notebook Output:** Inline animation rendering with Jupyter Notebook

**7. Conclusion**

This challenge successfully demonstrates how a single-layer perceptron can learn a linearly separable function like the OR gate using simple weight updates. The visualization illustrates each stage of learning, providing insight into how the decision boundary evolves and converges over time.