🎓 Logistic Regression – Part 1: The Perceptron Trick

This video lays the **foundation** for understanding **Logistic Regression** by introducing the **Perceptron Trick** — a conceptual stepping stone from linear models to classification.

➢ What is Logistic Regression?

- **Logistic Regression** is a **classification algorithm**, not a regression algorithm.
- It predicts **probabilities** (values between 0 and 1) using the **logistic (sigmoid) function**.
- Typically used for **binary classification** (e.g., spam vs not spam, 0 vs 1).

➢ Prerequisite for Logistic Regression

Before diving into logistic regression, it helps to understand:

1. **Linear regression** – predicting a continuous value.
2. **Decision boundaries** – separating regions of different classes in a feature space.
3. **Dot product** – helps measure the alignment of vectors (used in classification).

➢ The Perceptron Trick

🧠 What Is the Perceptron Trick?

The **Perceptron Algorithm** is a simple method for binary classification.

- You start with random weights (just like in logistic regression).
- For each sample, if it is **misclassified**, you **adjust the weights**.
- This pushes the decision boundary in the right direction.

📐 Intuition:

- If your model says “class 0” but the true label is “class 1”, then move the weights in the direction of the data point.
- If your model says “class 1” but it should be “class 0”, move the weights **away** from the data point.

This is an **iterative correction process**, very similar to what we do in logistic regression but with a **step function** instead of a sigmoid.

➢ How to Label Regions?

To classify, we define **regions** using the **decision boundary**:

- Decision boundary: **$w^T x + b = 0$**
- Everything **above** this line is class 1; everything **below** is class 0.

This is **very geometric** — you're drawing a line (or hyperplane) to divide your data.

➢ Transformations

Sometimes, data isn't linearly separable in 2D.

So, we can apply **feature transformations**, like:

- Polynomial features
- Interactions between features

This allows us to convert non-linear separability into **linear separability** in a higher-dimensional space (like in SVM or kernel methods).

➢ The Perceptron Algorithm

Here's the core idea of the **Perceptron Algorithm** (simplified steps):

1. Initialize weights to small random values.
2. For each training sample:
    1. Predict using current weights.
    2. If prediction is incorrect:

$$w = w + \Delta w$$

where:

$$\Delta w = \alpha \cdot (y_{\text{true}} - y_{\text{pred}}) \cdot x$$

and α is the learning rate.

3. Repeat until convergence or maximum epochs.

➢ Simplified Algorithm

The video also simplifies this into easy-to-understand logic for beginners:

- Loop through data.
- For every point, if it's wrongly classified, shift the weights toward (or away from) it.
- Keep repeating until the decision boundary correctly classifies all (or most) data.

🔑 Summary Takeaways

Concept	Explanation
Logistic Regression	Classification algorithm predicting probabilities using sigmoid
Perceptron Trick	Method to iteratively correct weights to classify data
Decision Boundary	Line/hyperplane dividing class 0 and class 1
Weight Updates	Based on whether sample is classified correctly
Transformations	Feature engineering to allow linear separation in higher dimensions

---