Polynomial Features in Logistic Regression | Non-Linear Logistic Regression

---

➢ Polynomial Features in Logistic Regression

- **Why:** Standard logistic regression models a **linear decision boundary**.
- **Problem:** Many real-world problems have **non-linear relationships** between features and target.
- **Solution:** Create **polynomial features** (e.g. x^2, x1x2, x^3) from the original features to allow the model to fit **curved decision boundaries**.

➢ How It Works

1. **Transform inputs**:
    - From x1,x2 → generate x_1^2, x_2^2, x_1 x_2, etc.
    - The degree of the polynomial controls complexity.
2. Apply **logistic regression** on these transformed features.
3. This lets the model separate classes that are not linearly separable in the original feature space.

➢ Pros & Cons

✅ **Pros:**

- Captures non-linear relationships without changing the logistic regression algorithm.
- Can dramatically improve performance on complex datasets.

⚠️ **Cons:**

- Higher-degree polynomials → risk of **overfitting**.
- Increases computation time and number of parameters.

➢ Implementation Tip

In Python:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
Then fit logistic regression on X_poly.
```
