Minor in AI

Mastering Polynomial Regression

1 The Curious Case of Plant Growth

Real-World Motivation

Imagine tracking a bamboo plant's growth over 10 days. Initially it grows slowly (1cm/day), then suddenly shoots up to 185cm by day 10. A linear model would draw a straight line through these points, but real-world data often curves and twists. This is where polynomial regression shines - it helps us model nature's curves!

Observed Data

Day	Growth (cm)
1	1
2	2
3	3
4	5
5	15
6	34
7	48
8	70
9	136
10	185

2 Mathematical Foundation

Understanding the mathematics behind regression gives us the power to model and predict relationships in data. Regression is a statistical process for estimating the relationships among variables. When we move beyond simple linear regression to higher-order polynomials, we can capture more complex, curved patterns in the data. This expanded foundation explains how each additional term (quadratic, cubic, etc.) adds flexibility to our models while also discussing the trade-offs involved.

2.1 Regression Evolution

- **Linear Regression:**

$$y = \beta_0 + \beta_1 x$$

  - β_0 (Intercept): Represents the expected value of y when $x = 0$. It is the starting point of the regression line.
  - β_1 (Slope): Indicates the constant rate of change in y for each unit change in x.
  - *Interpretation:* This model assumes a straight-line relationship between x and y. Every change in x results in a fixed change in y.

- **Quadratic Regression:**

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

  - β_2 (Curvature): Determines the degree of curvature. A positive value of β_2 creates a U-shaped curve, while a negative value creates an inverted U-shape.

– *Interpretation:* The inclusion of the x^2 term allows the model to capture acceleration or deceleration in the relationship. It is useful in scenarios where the rate of change of y itself changes with x, such as in growth spurts or braking distances.

- **Cubic Regression:**

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

– β_3 (Inflection): Adds the capability to model inflection points where the curvature changes direction. This allows the relationship to bend in an S-shaped pattern.

– *Interpretation:* The cubic model is ideal for capturing more complex patterns with peaks and valleys. It is applicable in contexts such as economic cycles or temperature changes where the trend reverses after certain critical points.

Real-World Analogy

Imagine driving a car:
Linear: Constant speed (e.g., 60 mph continuously)
Quadratic: Accelerating or braking steadily
Cubic: Complex maneuvers (accelerate, brake hard, then cruise)

2.2 Formula Interpretation

Each term in the regression formulas plays a specific role in modeling the relationship between the independent variable x and the dependent variable y.

- **Base Formula:**

$$y = \beta_0 + \beta_1 x$$

– *Interpretation:* y starts at the baseline value β_0 (when $x = 0$) and changes linearly with x at a rate of β_1 per unit change in x.

- **With x^2 Term:**

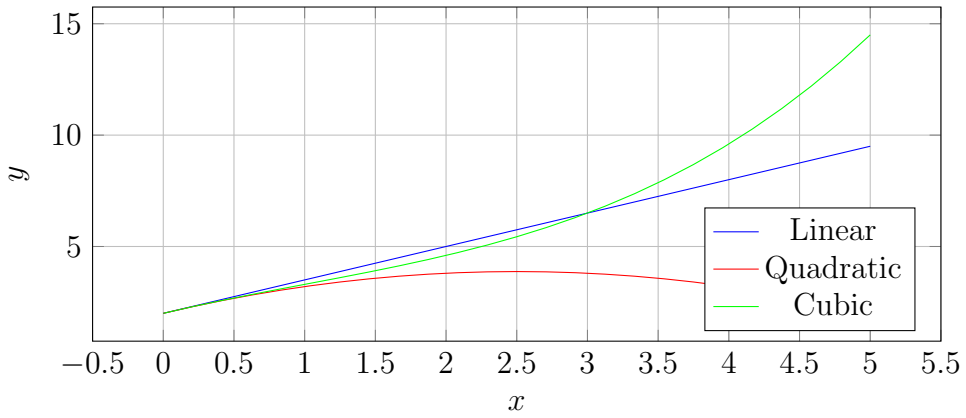
$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

– *Interpretation:* The quadratic term $\beta_2 x^2$ allows the model to capture acceleration (if $\beta_2 > 0$) or deceleration (if $\beta_2 < 0$) in the growth of y.

- **With x^3 Term:**

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

– *Interpretation:* The cubic term $\beta_3 x^3$ introduces flexibility to capture inflection points where the direction of the curve changes. This makes it possible to model S-shaped or multi-turn relationships.



Concept

Polynomial regression maintains *linearity in parameters* while modeling non-linear relationships through feature engineering.
Polynomial regression extends linear regression by adding powers of the independent variable (x^2, x^3, etc.), allowing us to model non-linear relationships while using linear algebra techniques.

3 Hands-on

Code

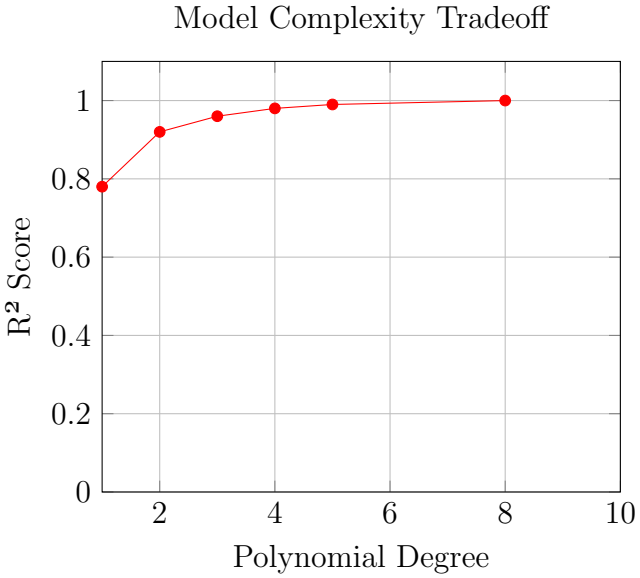
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score

# Data preparation
days = np.array([1,2,3,4,5,6,7,8,9,10]).reshape(-1,1)
growth = np.array([1,2,3,5,15,34,48,70,136,185])

# Polynomial transformation
poly = PolynomialFeatures(degree=3)
days_poly = poly.fit_transform(days)

# Model training
model = LinearRegression()
model.fit(days_poly, growth)

# Evaluation
predicted = model.predict(days_poly)
r2 = r2_score(growth, predicted)
print(f"R2 Score: {r2:.2f}")
print(f"Coefficients: {model.coef_}")
```

Understanding Code

In practical implementations, polynomial regression is performed using tools that handle feature transformation and model fitting. Below are key components often found in code:

- `PolynomialFeatures`: A tool (for example, from `scikit-learn`) that generates polynomial terms up to a specified degree from the input data. For instance, with a degree of 2, it produces both x and x^2 features.
- `fit_transform`: A method that fits the polynomial transformer to the data and then transforms the data into polynomial features. This prepares the dataset for regression analysis.
- `coef_`: An attribute of the fitted regression model that contains the estimated coefficients $\beta_1, \beta_2, \ldots$ corresponding to the polynomial terms.
- **R^2 Score**: A metric that measures the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 score of 1 indicates a perfect fit, while an R^2 score of 0 indicates no explanatory power.

4 Model Selection Strategy



4.1 Model Fitting

- **Underfitting (Degree=1)**: A linear model (degree=1) oversimplifies complex patterns, failing to capture curvature in data. It shows poor performance on both training and test sets due to high bias. Example: A straight line attempting to model exponential growth.
- **Optimal Fit (Degree=3-4)**: Balances model flexibility with generalization, capturing true trends without memorizing noise. Achieves high R^2 on training data while maintaining good test performance. Example: A smooth curve following natural growth patterns.
- **Overfitting (Degree=8)**: Excessively complex models (high degrees) fit training noise as "patterns," creating erratic curves. Perfect training accuracy (R^2=1) but fails catastrophically on new data. Example: A wiggly curve passing through every data point precisely.

Alarming

Higher degrees create perfect training fit but may fail with new data. Always validate!

5 Data Splitting Strategy

The data being divided into three segments: 70% for training, 15% for validation, and 15% for testing. The training set is used to build the model, the validation set to fine-tune it, and the test set to evaluate its performance.

6 Key Takeaways

- **Polynomial Regression:** The model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$$

  is used to model non-linear relationships by extending the linear model with higher-order terms.

- **Feature Engineering:** Polynomial regression is an example of feature engineering, where new features are created from the original data to capture complex patterns.

- **Model Fit (R-squared):** The R^2 score is crucial for evaluating how well the model explains the variability of the data.

- **Overfitting:** Increasing the degree of the polynomial can improve the fit on training data, but it also increases the risk of overfitting. Overfitting happens when the model learns the noise in the training data instead of the underlying pattern.

- **Model Validation:** It is essential to validate models using techniques such as cross-validation and test sets to ensure that they generalize well to unseen data.

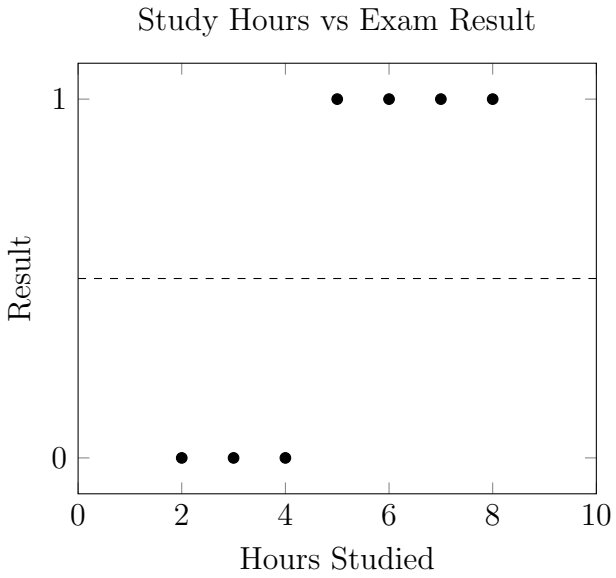
- **Appropriate Algorithm Choice:** Regression models are suited for continuous outcomes. For categorical outcomes, classification algorithms with different techniques and evaluation metrics are necessary.

References

- Colab - Code File
- Spreadsheet - Regression Errors

7 Student Performance Prediction

A new challenge:
Predict if students pass(1)/fail(0) based on study hours. Linear regression fails here - it predicts continuous values, not categories.



This leads us to...

The Classification Frontier

Teaser

When outcomes are categories (pass/fail, spam/not spam), we need classification algorithms like logistic regression - coming next!

Did You Know?

Polynomial regression was used by Gauss in 1809 for celestial mechanics!