The Mathematics of Polynomial Regression

Hom Bahrani

2025

Abstract

This document provides a comprehensive mathematical treatment of polynomial regression, focusing on the underlying theory, feature transformation, gradient descent optimization, and regularization techniques. We derive the necessary equations, analyze convergence properties, and explore the bias-variance tradeoff for polynomial models of varying degrees. The document connects statistical and geometric interpretations while providing practical insights for model selection and implementation.

Contents

1 Introduction to Polynomial Regression

Polynomial regression extends linear regression by modeling the relationship between a dependent variable and one or more independent variables using

polynomial functions. It allows us to capture non-linear relationships while still benefiting from the analytical framework of linear regression.

1.1 Mathematical Formulation

In standard linear regression, we model the relationship between features and the target as:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

Polynomial regression extends this by introducing higher-order terms. For a single feature x, a polynomial regression of degree d has the form:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_d x^d \tag{2}$$

For multiple features, polynomial regression includes not only powers of individual features but also interaction terms. For instance, with two features x_1 and x_2 and degree $d = 2$, the model becomes:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 \tag{3}$$

1.2 Relationship to Linear Regression

A key insight is that polynomial regression is still a form of linear regression from the perspective of the parameters θ. By defining new features as transformations of the original features (e.g., $z_1 = x_1$, $z_2 = x_2$, $z_3 = x_1^2$, etc.), we can express polynomial regression as:

$$h_\theta(x) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \cdots + \theta_m z_m \tag{4}$$

This means we can leverage the existing machinery of linear regression once we transform our feature space.

1.3 Polynomial Feature Expansion

For a feature vector $x \in \mathbb{R}^n$ and polynomial degree d, the expanded feature vector will include all possible combinations of features raised to powers that sum to at most d.

The number of features after polynomial expansion is given by the binomial coefficient:

$$\binom{n+d}{d} = \frac{(n+d)!}{n! \cdot d!} \tag{5}$$

where n is the number of original features.

For example, with 3 original features and degree 2, the expanded feature space would have $\binom{3+2}{2} = \binom{5}{2} = 10$ features.

1.4 Matrix Formulation

For a dataset with m examples and n original features, after polynomial expansion to degree d, we can express the problem in matrix form:

$$X_{poly} = \Phi(X) \tag{6}$$

where Φ is the feature transformation function that maps the original $m \times n$ feature matrix X to the expanded $m \times k$ polynomial feature matrix X_{poly}, where $k = \binom{n+d}{d}$.

The hypothesis function can then be written as:

$$h_\theta = X_{poly}\theta \tag{7}$$

2 Cost Functions for Polynomial Regression

The cost functions used in polynomial regression are the same as those used in linear regression. The difference lies in the feature representation, not in the objective function.

2.1 Mean Squared Error (MSE)

The most common cost function for polynomial regression is the Mean Squared Error (MSE):

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{8}$$

In matrix notation:

$$J(\theta) = \frac{1}{2m}(X_{poly}\theta - y)^T(X_{poly}\theta - y) \tag{9}$$

Expanding the matrix form:

$$(X_{poly}\theta - y)^T(X_{poly}\theta - y) = (X_{poly}\theta)^T(X_{poly}\theta) - (X_{poly}\theta)^T y - y^T(X_{poly}\theta) + y^T y \tag{10}$$

$$= \theta^T X_{poly}^T X_{poly}\theta - 2\theta^T X_{poly}^T y + y^T y \tag{11}$$

2.2 Mean Absolute Error (MAE)

An alternative cost function is the Mean Absolute Error (MAE):

$$J_{MAE}(\theta) = \frac{1}{m}\sum_{i=1}^{m}|h_\theta(x^{(i)}) - y^{(i)}| \tag{12}$$

2.3 Huber Loss

Huber loss combines the best properties of MSE and MAE:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases} \tag{13}$$

where $a = h_\theta(x^{(i)}) - y^{(i)}$ is the residual and δ is a hyperparameter.

2.4 Probabilistic Interpretation

Similar to linear regression, polynomial regression can be derived from a probabilistic perspective. If we assume:

$$y^{(i)} = \theta^T \phi(x^{(i)}) + \epsilon^{(i)} \tag{14}$$

where $\phi(x^{(i)})$ is the polynomial transformation of $x^{(i)}$ and $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise, then:

$$p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T \phi(x^{(i)}))^2}{2\sigma^2}\right) \tag{15}$$

Taking the negative log-likelihood and dropping constant terms, we get:

$$-\log L(\theta) \propto \sum_{i=1}^{m} (y^{(i)} - \theta^T \phi(x^{(i)}))^2 \tag{16}$$

This is equivalent to the MSE cost function. Thus, minimizing the MSE is equivalent to maximizing the likelihood under Gaussian noise assumptions.

3 Analytical Solution to Polynomial Regression

3.1 Normal Equation

For MSE, we can find the optimal parameters by setting the gradient to zero:

$$\nabla_\theta J(\theta) = \frac{1}{m} X_{poly}^T (X_{poly}\theta - y) = 0 \tag{17}$$

Solving for θ:

$$X_{poly}^T X_{poly} \theta = X_{poly}^T y \tag{18}$$

If $X_{poly}^T X_{poly}$ is invertible, the solution is:

$$\theta = (X_{poly}^T X_{poly})^{-1} X_{poly}^T y \tag{19}$$

This is known as the Normal Equation. It provides an exact solution without requiring iteration, but can be computationally expensive for high-degree polynomials due to the increased dimensionality and potential for numerical instability.

3.2 Limitations of the Analytical Solution

The analytical solution has several limitations:

- Computing $(X_{poly}^T X_{poly})^{-1}$ has time complexity $O(k^3)$, where $k = \binom{n+d}{d}$ is the number of features after polynomial expansion. For high-degree polynomials, this becomes prohibitively expensive.
- For large k, the inversion becomes computationally challenging and numerically unstable.
- If $X_{poly}^T X_{poly}$ is singular (non-invertible), we face the problem of multicollinearity, which is common in polynomial regression due to the correlation between different powers of the same feature.

4 Gradient Descent Optimization

4.1 The Gradient Descent Algorithm

The core idea of gradient descent is to iteratively update the parameters in the direction of steepest descent (negative gradient):

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{20}$$

where α is the learning rate.

4.2 Derivation of the Gradient for MSE

Let's derive the gradient of the MSE cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{21}$$

The partial derivative with respect to θ_j is:

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 && (22)\\
&= \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)}) - y^{(i)})^2 && (23)\\
&= \frac{1}{2m} \sum_{i=1}^{m} 2(h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)}) - y^{(i)}) && (24)\\
&= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)})) && (25)\\
& && (26)
\end{aligned}$$

Since $h_\theta(x^{(i)}) = \theta^T \phi(x^{(i)}) = \sum_{k=0}^{p} \theta_k \phi_k(x^{(i)})$, we have:

$$\frac{\partial h_\theta(x^{(i)})}{\partial \theta_j} = \phi_j(x^{(i)}) \tag{27}$$

Therefore:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \phi_j(x^{(i)}) \tag{28}$$

In vector form, the gradient is:

$$\nabla_\theta J(\theta) = \frac{1}{m} X_{poly}^T (X_{poly}\theta - y) \tag{29}$$

4.3 Gradient Descent Variants

4.3.1 Batch Gradient Descent

The standard gradient descent algorithm updates parameters using the entire dataset:

$$\theta := \theta - \alpha \nabla_\theta J(\theta) \tag{30}$$

4.3.2 Stochastic Gradient Descent (SGD)

SGD updates parameters using a single randomly selected example:

$$\theta := \theta - \alpha \nabla_\theta J_i(\theta) \tag{31}$$

where $J_i(\theta) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ is the cost for a single example.

4.3.3 Mini-Batch Gradient Descent

Mini-batch gradient descent updates parameters using a small random subset (mini-batch) of the data:

$$\theta := \theta - \alpha \nabla_\theta J_B(\theta) \tag{32}$$

where $J_B(\theta)$ is the cost for a mini-batch B.

4.4 Convergence Analysis

4.4.1 Learning Rate

The learning rate α determines the step size in each iteration:

- If α is too small, convergence will be slow.
- If α is too large, the algorithm may oscillate or diverge.

For convex functions like MSE, there exists a theoretical upper bound on the learning rate to ensure convergence. If λ_{max} is the largest eigenvalue of the Hessian matrix $\nabla^2 J(\theta)$, then convergence is guaranteed if:

$$0 < \alpha < \frac{2}{\lambda_{max}} \tag{33}$$

For polynomial regression, the Hessian is:

$$\nabla^2 J(\theta) = \frac{1}{m} X_{poly}^T X_{poly} \tag{34}$$

4.4.2 Convergence Rate

For a convex differentiable function with Lipschitz continuous gradients, batch gradient descent converges at a rate of $O(1/k)$, where k is the number of iterations. That is:

$$J(\theta^{(k)}) - J(\theta^*) \leq \frac{C}{k} \tag{35}$$

where θ^* is the optimal parameter and C is a constant.

5 Feature Normalization

Feature normalization is particularly crucial for polynomial regression because higher-degree terms can have dramatically different scales.

5.1 Standardization

Standardization transforms features to have zero mean and unit variance:

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{36}$$

where μ is the mean and σ is the standard deviation of the feature.

5.2 Min-Max Scaling

Min-max scaling transforms features to a specific range, typically $[0, 1]$:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{37}$$

5.3 Mathematical Justification

Consider a polynomial regression with a single feature x. After expansion to degree d, we have features $x, x^2, x^3, \ldots, x^d$. If x has a range of $[1, 10]$, then x^d has a range of $[1, 10^d]$. This exponential increase in scale leads to several issues:

- The condition number of the matrix $X_{poly}^T X_{poly}$ increases dramatically, leading to numerical instability in the analytical solution.
- In gradient descent, the partial derivatives with respect to higher-degree terms are much larger, causing oscillations and requiring a much smaller learning rate.

Normalization addresses these issues by bringing all features to a similar scale. For polynomial features, it's generally better to normalize the original features first and then generate polynomial terms, rather than normalizing after expansion.

6 Regularization in Polynomial Regression

Polynomial regression is particularly prone to overfitting as the degree increases. Regularization techniques help mitigate this issue.

6.1 Ridge Regression (L2 Regularization)

Ridge regression adds a sum of squared parameter values to the cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{p} \theta_j^2 \tag{38}$$

The regularization term excludes the bias parameter θ_0. In matrix form:

$$J(\theta) = \frac{1}{2m} (X_{poly}\theta - y)^T (X_{poly}\theta - y) + \frac{\lambda}{2m} \theta_{-0}^T \theta_{-0} \tag{39}$$

where θ_{-0} denotes θ without the bias term.
The gradient becomes:

$$\nabla_\theta J(\theta) = \frac{1}{m} X_{poly}^T (X_{poly}\theta - y) + \frac{\lambda}{m} \theta' \tag{40}$$

where θ' is θ with the bias term set to zero.
The analytical solution with ridge regularization is:

$$\theta = (X_{poly}^T X_{poly} + \lambda I')^{-1} X_{poly}^T y \tag{41}$$

where I' is the identity matrix with the first diagonal element set to zero (to exclude regularization on the bias term).

6.2 Lasso Regression (L1 Regularization)

Lasso regression adds a sum of absolute parameter values:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{m} \sum_{j=1}^{p} |\theta_j| \tag{42}$$

The L1 penalty encourages sparsity (many parameters becoming exactly zero), which is particularly valuable in polynomial regression where many higher-order terms may be unnecessary.

6.3 Elastic Net

Elastic Net combines L1 and L2 penalties:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{m} \left(\alpha \sum_{j=1}^{p} |\theta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \theta_j^2 \right) \tag{43}$$

where $\alpha \in [0, 1]$ controls the balance between L1 and L2 regularization.

7 The Bias-Variance Tradeoff

The bias-variance tradeoff is particularly evident in polynomial regression, where the model complexity is directly related to the polynomial degree.

7.1 Mathematical Formulation

For a given model h and target function f, the expected test error at a point x can be decomposed as:

$$\mathbb{E}[(y - h(x))^2] = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error} \tag{44}$$

where:

$$\text{Bias}^2 = (f(x) - \mathbb{E}[h(x)])^2 \tag{45}$$

$$\text{Variance} = \mathbb{E}[(h(x) - \mathbb{E}[h(x)])^2] \tag{46}$$

$$\text{Irreducible Error} = \sigma^2 \tag{47}$$

7.2 Bias-Variance in Polynomial Regression

In polynomial regression, as the degree d increases:

- Bias decreases: Higher-degree polynomials can better approximate complex functions.
- Variance increases: The model becomes more sensitive to fluctuations in the training data.

7.3 Optimal Degree Selection

The optimal polynomial degree d^* minimizes the expected test error:

$$d^* = \arg\min_d \mathbb{E}[(y - h_d(x))^2] \tag{48}$$

This can be estimated through techniques like cross-validation.

8 Model Selection Techniques

8.1 Cross-Validation

8.1.1 K-Fold Cross-Validation

The dataset is divided into K folds. For each fold k:

1. Train the model on all folds except the k-th.
2. Evaluate the model on the k-th fold.

The cross-validation error is the average error across all folds:

$$CV(d) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|F_k|} \sum_{i \in F_k} (y^{(i)} - h_d^{-k}(x^{(i)}))^2 \tag{49}$$

where F_k is the set of indices in the k-th fold and h_d^{-k} is the model trained on all folds except the k-th with polynomial degree d.

8.1.2 Leave-One-Out Cross-Validation (LOOCV)

LOOCV is a special case of K-fold cross-validation where $K = m$ (the number of examples). It provides an almost unbiased estimate of the test error but can be computationally expensive.

8.2 Information Criteria

8.2.1 Akaike Information Criterion (AIC)

AIC balances the goodness of fit with model complexity:

$$\text{AIC} = 2k - 2\ln(L) \tag{50}$$

where k is the number of parameters (related to the polynomial degree) and L is the maximum likelihood.

8.2.2 Bayesian Information Criterion (BIC)

BIC is similar to AIC but penalizes model complexity more strongly:

$$\text{BIC} = \ln(m)k - 2\ln(L) \tag{51}$$

where m is the number of examples.

9 Statistical Properties of Polynomial Regression

9.1 Unbiasedness of OLS Estimator

Under the standard assumptions, the Ordinary Least Squares (OLS) estimator $\hat{\theta} = (X_{poly}^T X_{poly})^{-1} X_{poly}^T y$ is unbiased:

$$\mathbb{E}[\hat{\theta}] = \theta \tag{52}$$

9.2 Variance of OLS Estimator

The variance-covariance matrix of the OLS estimator is:

$$\text{Var}(\hat{\theta}) = \sigma^2 (X_{poly}^T X_{poly})^{-1} \tag{53}$$

where σ^2 is the variance of the error term ϵ.

As the polynomial degree increases, the diagonal elements of $(X_{poly}^T X_{poly})^{-1}$ tend to increase, leading to higher variance in the parameter estimates.

9.3 Confidence Intervals for Parameters

Assuming errors are normally distributed, we can construct confidence intervals for the parameters. For the j-th parameter:

$$\hat{\theta}_j \pm t_{n-p,1-\alpha/2} \cdot \hat{\sigma} \sqrt{(X_{poly}^T X_{poly})_{jj}^{-1}} \tag{54}$$

where:

- $t_{n-p,1-\alpha/2}$ is the critical value from the t-distribution with $n-p$ degrees of freedom
- $\hat{\sigma}$ is the estimated standard deviation of the error term
- $(X_{poly}^T X_{poly})_{jj}^{-1}$ is the j-th diagonal element of $(X_{poly}^T X_{poly})^{-1}$

9.4 Goodness-of-Fit Measures

9.4.1 Coefficient of Determination (R^2)

R^2 measures the proportion of variance in the dependent variable that is predictable from the independent variables:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y^{(i)} - \hat{y}^{(i)})^2}{\sum_i (y^{(i)} - \bar{y})^2} \tag{55}$$

For polynomial regression, R^2 typically increases with the degree, potentially reaching 1 if the degree equals $m - 1$. This highlights the need for adjusted measures.

9.4.2 Adjusted R^2

Adjusted R^2 accounts for the number of predictors:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{56}$$

where n is the number of samples and p is the number of predictors (excluding the constant term).

10 Computational Implementation of Polynomial Regression

10.1 Efficient Feature Generation

Generating polynomial features can be computationally intensive. For a feature vector $x \in \mathbb{R}^n$ and degree d, a naive approach would generate $O(n^d)$ features. However, many of these are duplicates (e.g., $x_1 x_2$ and $x_2 x_1$).

An efficient approach is to generate only unique combinations. The total number of unique features is:

$$\binom{n + d}{d} = \frac{(n + d)!}{n! \cdot d!} \tag{57}$$

which grows polynomially with n and d, not exponentially.

10.2 Algorithm for Polynomial Regression

10.3 Converting Parameters Back to Original Scale

After training with normalized features, we need to convert the parameters back to the original scale. For each polynomial feature, the conversion depends on the specific combination of original features and their powers.

Algorithm 1 Polynomial Regression with Gradient Descent

Input: Training data (X, y), polynomial degree d, learning rate α, iterations T
Output: Model parameters θ
Generate polynomial features: $X_{poly} = \Phi(X)$
Normalize features: $X_{poly\_norm} = \frac{X_{poly} - \mu}{\sigma}$
Initialize $\theta = \vec{0}$
for $t = 1$ to T **do**
    $\hat{y} = X_{poly\_norm}\theta$
    $\nabla J = \frac{1}{m} X_{poly\_norm}^T (\hat{y} - y)$
    $\theta = \theta - \alpha \nabla J$
end for
Convert θ back to original scale
return θ

For a single feature transformed to a polynomial of degree d, the conversion is:

$$x_{norm}^d = \left(\frac{x - \mu}{\sigma} \right)^d = \frac{(x - \mu)^d}{\sigma^d} \tag{58}$$

Expanding using the binomial theorem:

$$x_{norm}^d = \frac{1}{\sigma^d} \sum_{i=0}^{d} \binom{d}{i} x^i (-\mu)^{d-i} \tag{59}$$

This means each normalized polynomial feature is a linear combination of powers of the original feature. Converting parameters back requires solving a system of linear equations.

10.4 Numerical Stability Considerations

When implementing polynomial regression, several numerical issues can arise:

- Ill-conditioned matrices: As the polynomial degree increases, $X_{poly}^T X_{poly}$ becomes increasingly ill-conditioned, leading to numerical instabilities in the analytical solution. Using QR decomposition or SVD can mitigate this.
- Overflow/underflow: High-degree polynomial terms can cause numerical overflow or underflow. Proper normalization of the original features before generating polynomial terms helps prevent this.
- Multicollinearity: Polynomial features are often highly correlated, leading to unstable parameter estimates. Regularization techniques like Ridge regression can address this issue.

11 Conclusion and Advanced Topics

Polynomial regression serves as a bridge between simple linear models and more complex non-linear approaches. While we've covered the core mathematics, several advanced topics extend from here:

- Spline Regression: Extends polynomial regression by fitting piecewise polynomials with constraints on smoothness at the knots.
- Orthogonal Polynomials: Uses basis functions that are orthogonal to each other, improving numerical stability.
- Kernel Methods: Implicitly maps features to a high-dimensional space without explicitly computing the polynomial features.
- Regularization Path Algorithms: Computes the entire solution path for different regularization parameters efficiently.
- Local Polynomial Regression: Fits polynomials to local neighborhoods of the data, allowing for more flexibility.

Understanding the mathematical foundations of polynomial regression provides insight into the bias-variance tradeoff and the importance of model selection in machine learning. It also serves as a foundation for understanding more complex non-linear regression techniques.