# Polynomial Regression

The previous two chapters have employed linear functions as models for our house sales data – a linear function is an appropriate model to use when a straight line fits your data best. We briefly introduced other types of functions that can be used as models for your data when a straight line isn’t the best fit.

In this chapter we introduce polynomial regression and polynomial functions. Polynomial functions are compelling models for *nonlinear data*; put simply, data that does not fit a straight line. For example, polynomial functions would be much better at modelling bell curves than a linear function would.

One of the most interesting things about this chapter is that we won’t be introducing new hypothesis functions, cost functions or gradient descent algorithms; if you need a bit of a break from all of the math, this is your chapter! Polynomial regression will leverage *absolutely all* of the machinery that we developed in multivariate linear regression.

Instead of developing algorithms for polynomial regression, we are going to learn about a technique to transform *data sets*. While the technique is quite simple, it will allow you to develop sophisticated predictive models for nonlinear data.

## 

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## Polynomial Functions

Polynomial functions are versatile data modelling tools. For your reference, figure x below presents three different degrees of polynomial functions, along with the definition of the function and a visual of the shape each function can notionally produce. While higher degree polynomials may be used for modelling, lower degree polynomials like the ones presented below are the most commonly used polynomial functions for machine learning.

Figure 1: Polynomial functions

|  |  |
| --- | --- |
| **Second Degree (Quadratic) Polynomial Functions**  f(x) = a + bx + cx2 |  |
| **Third Degree (Cubic) Polynomial Functions**  f(x) = a + bx + cx2 + dx3 |  |
| **Fourth Degree (Quartic) Polynomial Functions**  f(x) = a + bx + cx2 + dx3 + ex4 |  |

## Expressing Polynomial Functions in Data

As we mentioned at the beginning of this chapter, data transformation is the approach to implementing polynomial regression, not algorithm development. In this section we’ll show you how to transform your data in a way that allows a polynomial function to be *expressed as* a linear function in your data. Once the transformation is complete, you can reuse the multivariate linear regression algorithm we developed last chapter to train and make predictions on nonlinear data sets.

Let’s work through a concrete example to demonstrate how this technique works in practice. To start, we’ll define a simple house sales training set with three different features – number of bedrooms, number of bathrooms and house age.

Table 3: Training set with three features.

| **Bedrooms** | **Bathrooms** | **Age** | **House Price** |
| --- | --- | --- | --- |
| 3 | 1 | 3 | 223000 |
| 3 | 2 | 11 | 430000 |
| 5 | 3 | 22 | 900000 |
| 4 | 1 | 44 | 300000 |

In the previous chapter we developed the multivariate linear regression hypothesis function; the same hypothesis function will be used in polynomial regression. As a quick refresher, a multivariate linear regression hypothesis function with three features is defined as:

Where is our conventional constant value 1, is the number of bedrooms, is number of bathrooms and is the age of the house. As a linear function, it produces a straight line on a graph.

Let’s assume that a straight line is *not* what we’re looking for; instead, we discover that our housing data would be best modelled with a third degree (cubic) polynomial function. By quickly referring to figure x above, we know that the hypothesis function for a cubic polynomial is:

So how do we go about expressing this in our data exactly? Let’s make a quick side-by-side comparison of the multivariate linear hypothesis function and the cubic polynomial hypothesis function.

// multivariate linear hypothesis function

// cubic polynomial hypothesis function

You’ll notice that the two functions are nearly identical, except for the squared term and cubed term in the polynomial function (highlighted in red). The key insight is this: a *linear function* that uses *squared and cubed data* will output the exact same results as a *polynomial function* that squares and cubes the *original data*. We’re really just shifting *when* and *where* data gets squared or cubed.

We’ll add two new features to our training set, **Bathrooms2**and **Age3**. The Bathrooms2 feature squares the existing bathroom values, and the Age3 feature cubes the existing age values.

Table 3: Training set with three features.

| **Bedrooms** | **Bathrooms2** | **Age3** | **House Price** |
| --- | --- | --- | --- |
| 3 | 1 | 27 | 223000 |
| 3 | 4 | 1331 | 430000 |
| 5 | 9 | 10648 | 900000 |
| 4 | 1 | 85184 | 300000 |

After data exponentiation, our multivariate linear regression hypothesis function is effectively:

This linear function is *functionally equivalent* to a polynomial function that uses the original training data; both functions will exhibit the same behavior and compute identical values for h(x). This data exponentiation technique can be used to map any degree polynomial function – we just happened to use a third degree polynomial function as an example.

Exponentiation will most likely increase the range and size of your training data, so please ensure that you use feature scaling to normalize your data *before* training the univariate linear regression algorithm.

# Support Vector Machines

Placeholder for chapter content.

# K-Means Clustering

Placeholder for chapter content.

# Principal Component Analysis

Placeholder for chapter content.

# Neural Networks

Placeholder for chapter content.

# Anomaly Detection

Placeholder for chapter content.

# Recommender Systems

Placeholder for chapter content.

# Applying Machine Learning in Practice

Placeholder for chapter content.

# Applying Machine Learning at Scale

Placeholder for chapter content.

# Appendix A: Linear Functions Review

This section provides an overview of linear functions and includes a practical example that you can work through. By the end of this section you should understand what a linear function is and have the ability to graph a linear function.

A *linear function* is a simple mathematical function whose graph is a straight line. You would have certainly studied linear functions in high school, but in case it’s been a while this is a brief refresher on linear functions, which take the form:

y = ax + b

The value **a** is referred to as the slope, **b** is the intercept, **x** is the independent variable and **y** is the dependent variable.

Let’s run through a concrete example and graph a linear function where a = 5 and b = 2, so our linear function is:

y = f(x) = 5x + 2

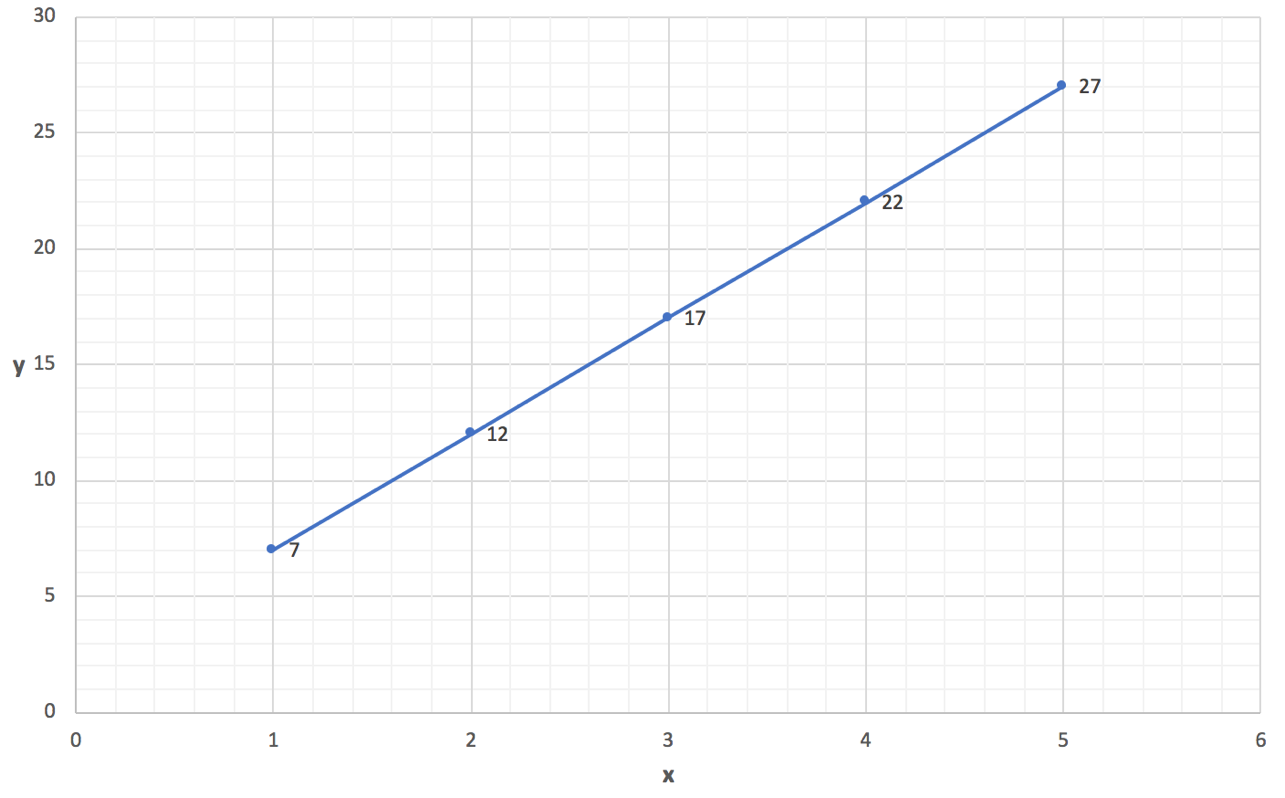
Table 1 presented below shows the calculation of the dependent variable y for various values of x, ranging from x=1 to x=5.

Table 6: Examples of running various values of x through linear function f(x) = 5x + 2

|  |  |  |
| --- | --- | --- |
| **x (Independent Variable)** | **f(x) = 5x + 2** | **y (Dependent Variable)** |
| 1 | f(1) = 5\*1 + 2 | 7 |
| 2 | f(2) = 5\*2 + 2 | 12 |
| 3 | f(3) = 5\*3 + 2 | 17 |
| 4 | f(4) = 5\*4 + 2 | 22 |
| 5 | f(5) = 5\*5 + 2 | 27 |

Now that we’ve calculated y for each value of x, we plot each of the (x,y) pairs on a graph and draw a line through each pair. Doing so produces a straight line on a graph, as shown in figure 8.

Figure 4: A graph of the linear function f(x) = 5x + 2



This simple example demonstrated the calculations to produce the (x,y) pairs and what the resulting linear function looks like when drawn on a graph.

# Appendix B: Matrix Algebra Review