Supervised Learning: Linear and Polynomial Regression

Abdelrahman Khaled

Machine Learning Research Cluster
German University in Cairo

January 22, 2019

Outline

Supervised Learning

We already vaguely know what supervised learning is and what it's supposed to do, however we still don't know how to formalize a problem to solve it using a supervised learning algorithm.
Previously we said that a data set A contains data points x and that each x contains a list of features and a label.

Supervised Learning: Notation

In order to make things easier to understand, let's introduce the data set A with the more well known standard notation.

Standard Notation

$A = (X, Y)$ for n data points where;

$X = \{x_1, x_2, \ldots, x_n\}$ is the set of all feature vectors for all data points and;
$Y = \{y_1, y_2, \ldots, y_n\}$ is the set of all labels for all data points.

Supervised Learning: Notation

In order to make things easier to understand, let's introduce the data set A with the more well known standard notation.

Standard Notation

$A = (X, Y)$ for n data points where;

$X = \{x_1, x_2, \ldots, x_n\}$ is the set of all feature vectors for all data points and;
$Y = \{y_1, y_2, \ldots, y_n\}$ is the set of all labels for all data points.

Using this notation we can better describe a supervised learning algorithm as trying to learn the function f that maps from the domain of X to the domain of Y.

$$f(x_i) = y_i$$

Regression Problems

Previously we differentiated between classification problems and regression problems. We said that classification learns to map inputs to a number of specific classes (discrete output) while regression aims to predict a real number (continuous output). So the function f that our program is trying to learn has the domain $\mathbb{R}$.

Normally we only have a small amount of the data needed to solve the problem. We call the data that we do have "The training set", and we use it to make a hypothesis h of what f should be.

Linear Regression

For data that is similar to the figure below, it is sufficient to fit the data with a single line h that approximates the actual function f.

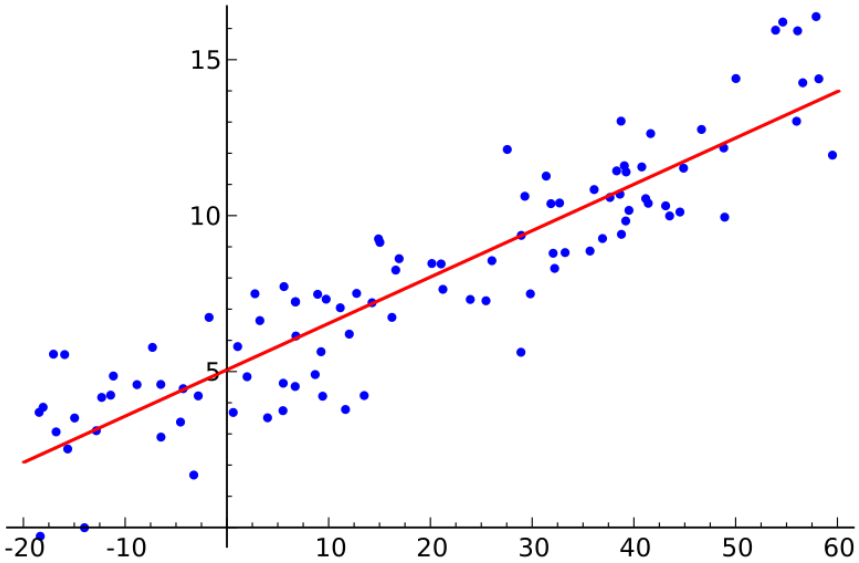

Figure: Data with a linearly increasing trend. *Image Source*

$$h(x_i) \approx y_i$$

Linear Regression

Since h is a line it can be represented using the straight line equation such that;

$$h(x_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im}$$

The idea is to choose values for β_0 through β_m so that h is as close as possible to f for the examples in the training set.

To do that we have to minimize the average error between $h(x_i)$ and y_i for all i and we can achieve that by minimizing the squared error. We call that the cost function.

$$C(B) = \frac{1}{2n} \sum_{i}^{n} (h(x_i) - y_i)^2$$

Linear Regression: Gradient Descent

Gradient descent computes the path of steepest descent on any surface. That means that if we're trying to find the minimum of the cost, all we have to do is perform gradient descent on $C(B)$ until convergence.
Gradients get the direction of steepest ascent for each variable individually through the partial derivative, so intuitively the negative of the gradient will be gradient descent.

Applying gradient descent onto our linear regression model gets us the following update rule:

$$\beta_j = \beta_j - \alpha \frac{\partial}{\partial \beta_j} C(B)$$

We don't need to calculate the derivative at all, because most machine learning libraries do the hard work for us.

Linear Regression: Code Example

live example

Polynomial Regression

Sometimes fitting a line to the data isn't enough, so we try to fit a curve.

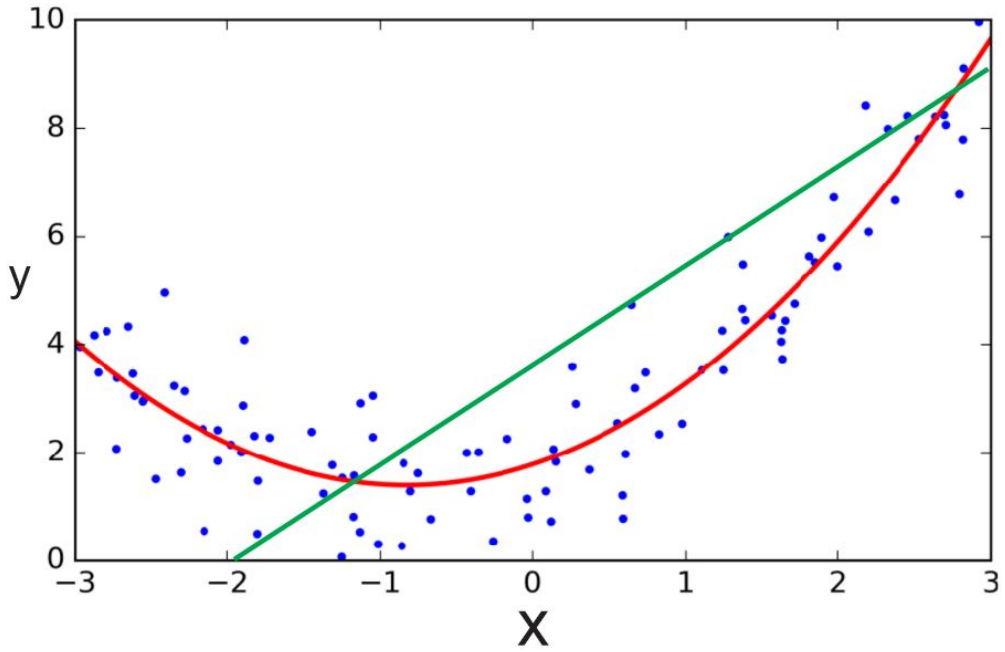

Figure: Data with a trend similar to a concave up parabola. *Image Source*

Polynomial Regression

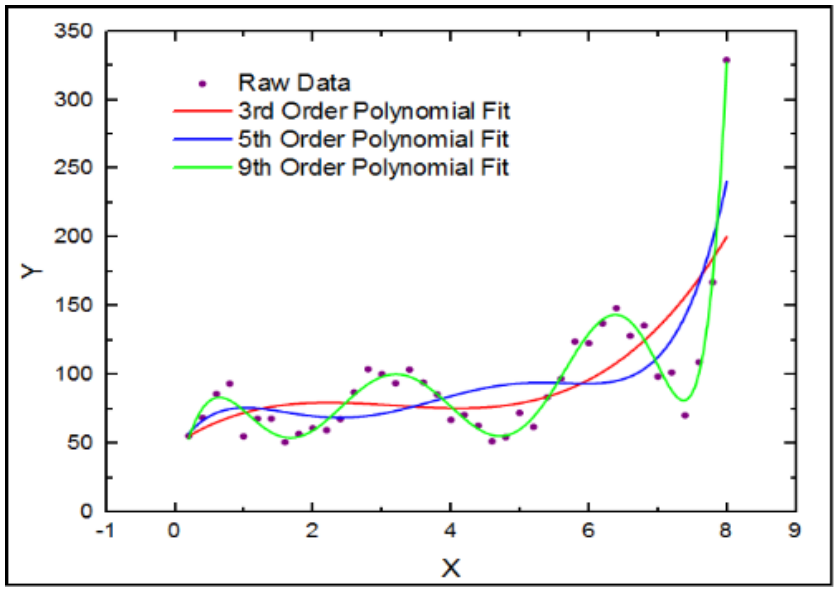

Figure: Fitting different degrees of curves onto a data set. *Image Source*

So if we choose to fit with a degree p curve, the hypothesis h (for one feature) changes to;

$$h(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p$$

Matrix Representation

So far we've considered each individual data point in a data set when computing the hypothesis or the cost functions.
Linear Algebra gives us matrix notation that makes it much easier for us to represent these groups of operations.

So, for a linear regressor with n datapoints and m features;

$$\underset{Y_{(n\times 1)}}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}} = \underset{[1]\cdot X_{(n\times(m+1))}}{\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{im} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{nm} \end{bmatrix}} \underset{B_{((m+1)\times 1)}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_j \\ \vdots \\ \beta_m \end{bmatrix}}$$

References

- *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Trevor Hastie, Robert Tibshirani, Jerome Friedman. Springer, 2013. **Section 3.2**