Machine Learning applied to Planetary Sciences

PTYS 595B/495B

Leon Palafox

https://leonpalafox.github.io/MLClass/

News of the day

The Arrival of Artificially Intelligent Beer

WRITTEN BY ALASDAIR ALLAN

29 August 2016 // 02:37 PM CET

http://motherboard.vice.com/en_uk/read/the-arrival-of-artificially-intelligent-beer

Last Class Recap – Logistic Regression

- Imagine you want to sell you car:
  - How much do you ask for it:
    - Mileage
    - Year
    - Color
    - Options
    - Condition

What do places like Edmunds.com do?

Model	Year	Brand	Price	Options	Condition	Mileage
Corvette	**1961**	**Chevrolette**	**100K**	**Standard**	**As New**	**100,000**
Corvette	**1961**	**Chevrolette**	**10K**	**Standard**	**Rust**	**100,000**
Corvette	**1961**	**Chevrolette**	**120K**	**Standard**	**Used**	**20,000**

Price = Year + Options + Condition + Mileage

This doesn't work

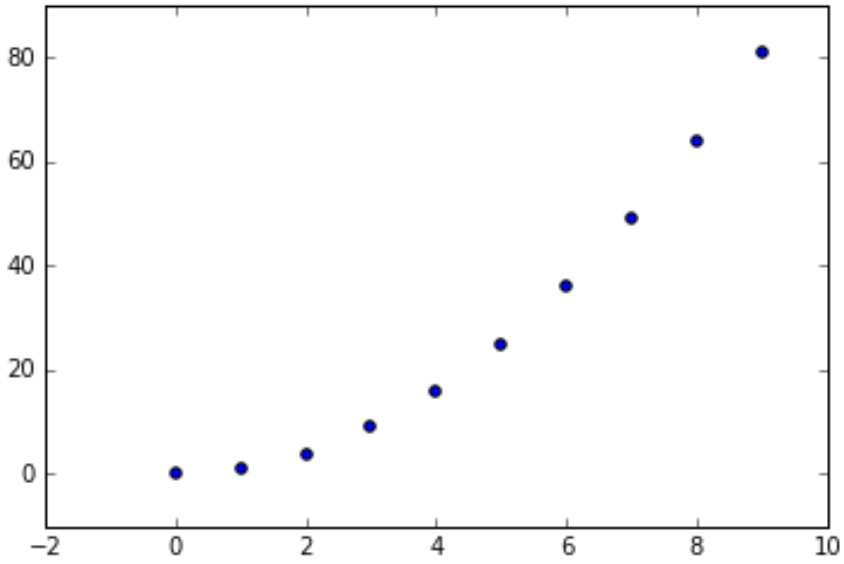
Price = A*Year+ B*Options + C*Condition+ D*Mileage

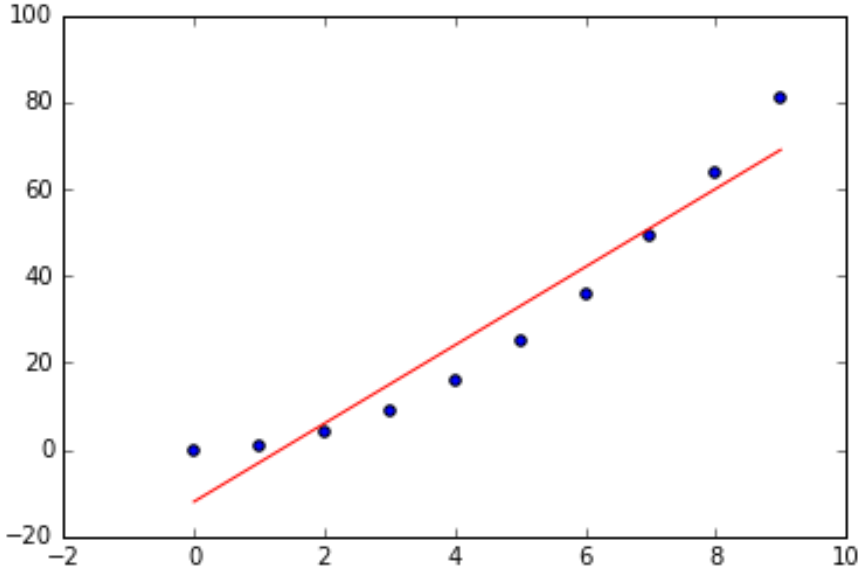
Matrix form

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix},$$

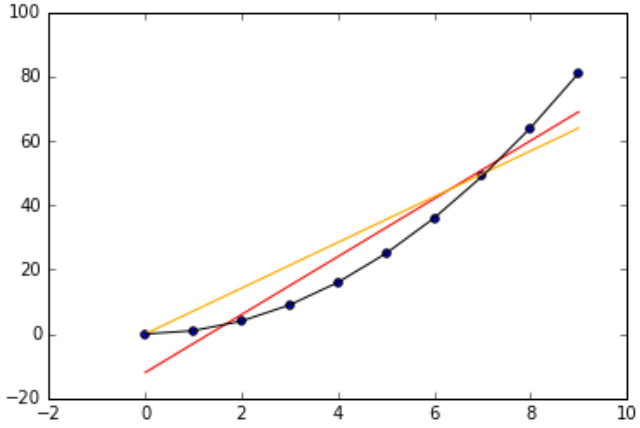
$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} \qquad \mathbf{y} = \mathbf{X}\beta$$

Feature Engineering



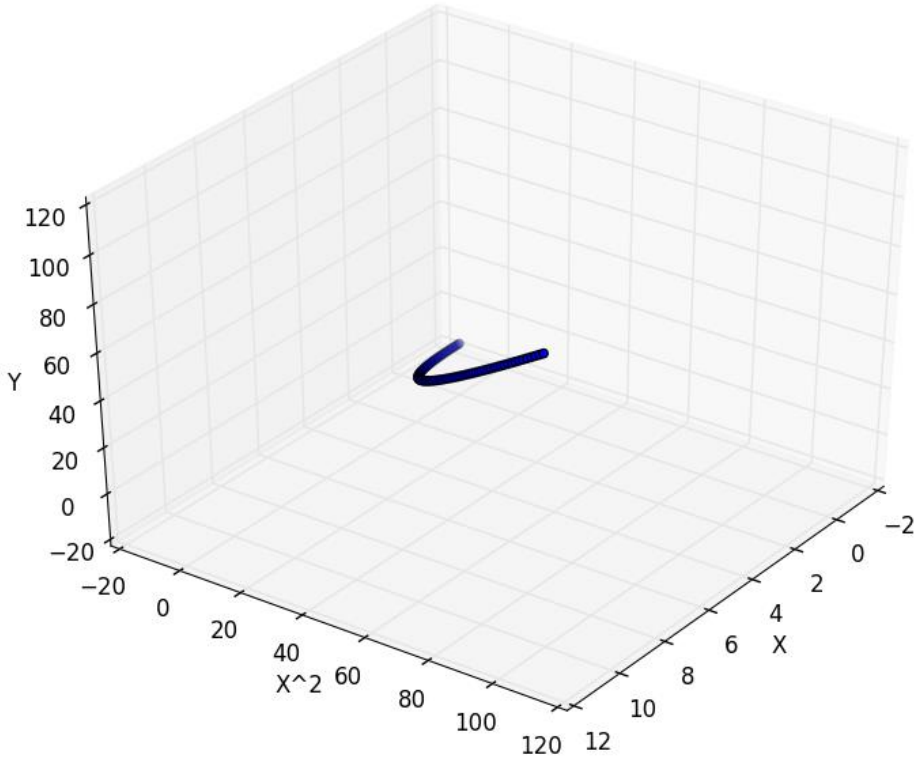


By adding an extra squared factor, we got a perfect fit



Math background

- By adding extra features, we move to higher dimensional spaces.



Optimization problem

- Optimize:

$$\mathbf{y} = \mathbf{X}\beta$$

- Score:

$$X\beta - y$$

$$(X\beta - y)^2$$

$$\frac{1}{2}(X\beta - y)^2$$

This is considered a cost!

Matrix Algebra approach

$$\frac{1}{2}(X\beta - y)^2 = \frac{1}{2}(X\beta - y)^T(X\beta - y)$$

$$= J(\beta)$$

$$\nabla_\beta J(\beta) = 0$$

Matrix Algebra Approach

$$\nabla_\beta J(\beta) = X^T X\beta - X^T y = 0$$

$$X^T X\beta = X^T y$$

$$\beta = (X^T X)^{-1} X^T y$$

Gradient descent

$$h_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

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

$$\boxed{\beta_{j+1} = \beta_j - \alpha \frac{\delta}{\delta \beta_j} J(\beta)}$$

Gradient descent

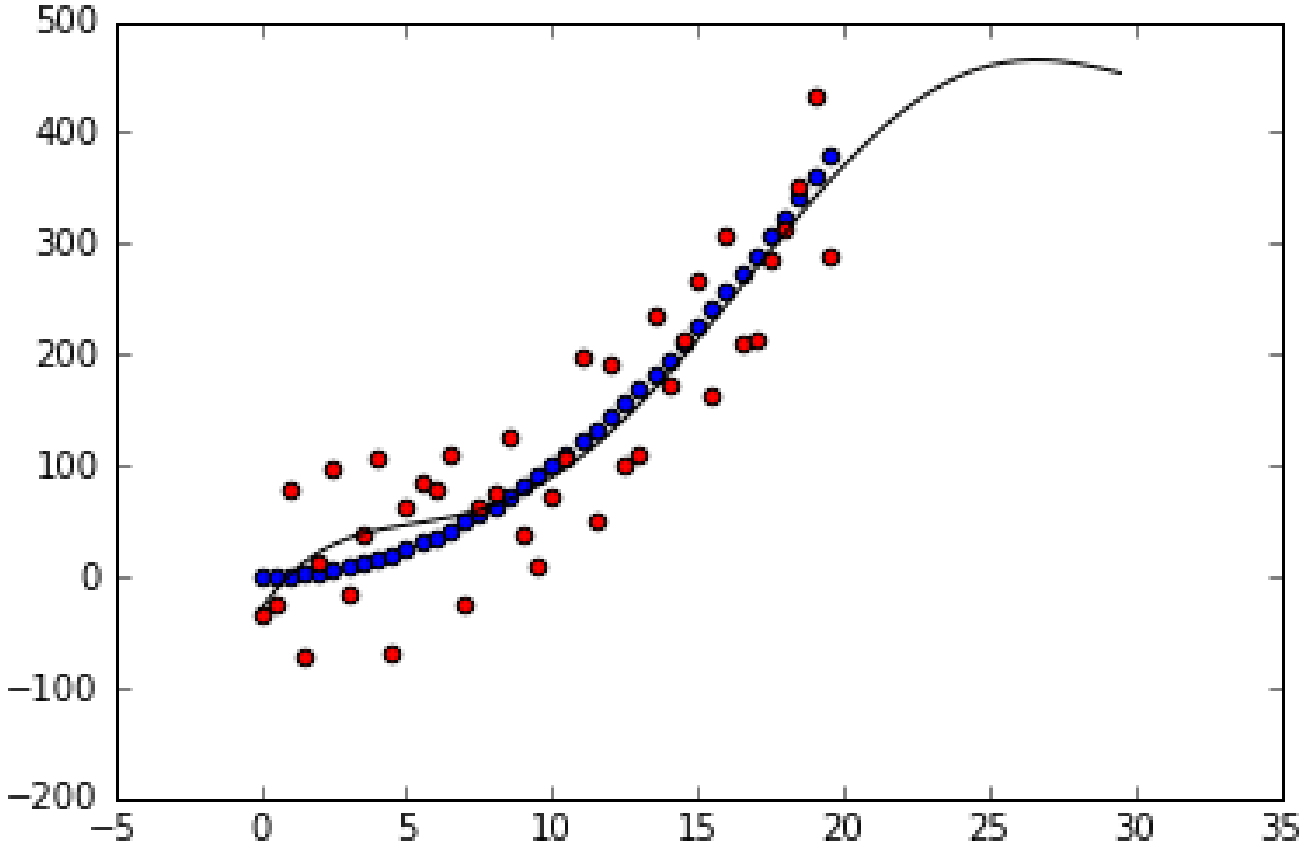
$$\beta_{j+1} = \beta_j - \alpha(y - h_\beta(x^{(i)}))x_j^{(i)}$$

What is the difference

- Large amount of data:
  - Gradient descent rules!
  - Matrix Algebra is a pain.
- Small datasets
  - Matrix algebra is way faster.

- In practice, most implementations use Gradient Descent, since datasets get large very fast.

How to control that polynomial problem?



Regularization

- Tries to keep the weights in check, by adding an extra penalty to the cost.

$$J(\beta) = \frac{1}{2}(y - h_\beta(x^{(i)}))^2 + \lambda \|\beta\|$$

$$J(\beta) = \frac{1}{2}(y - h_\beta(x^{(i)}))^2 + \alpha \|\beta\|$$

- We refer to this as ridge regression