Regression Algorithms

Xavier Morera

Helping developers understand and work with data

@xmorera www.xaviermorera.com / www.bigdatainc.org

$$SIN\ \alpha = COS\ (90 - \alpha)$$

$$\lim_{x \to 0} x^{b} \log_a x = 0, \quad (b > 0)$$

$$\frac{1}{x \to 0} \quad x \quad \ln a$$

$$K_1(t,s) = K(t,s)$$

$$Arc\ COS\ z = -i\ Ln(z + i\sqrt{1-z^2})$$

$$A = \frac{1}{2}\ ab$$

$$V = \frac{1}{3}\ \pi r^2 h$$

$$K_0(x,t) = K(x,t)$$

$$\varphi(x) = \lambda \int K(x, s)\, \varphi(s)\, ds + f(x)$$

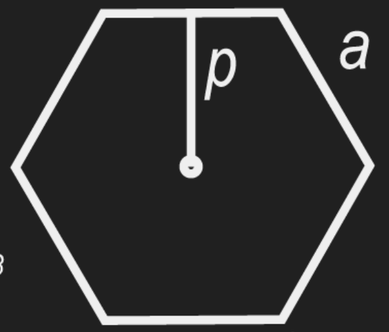
$$z = re^{i\varphi}$$

$$ctg\ \alpha + ctg\ \beta = \frac{SIN\ (\alpha + \beta)}{SIN\ \alpha\ SIN\ \beta}$$

$$tg\ \alpha + tg\ \beta = \frac{SIN\ (\alpha + \beta)}{COS\ \alpha\ COS\ \beta}$$

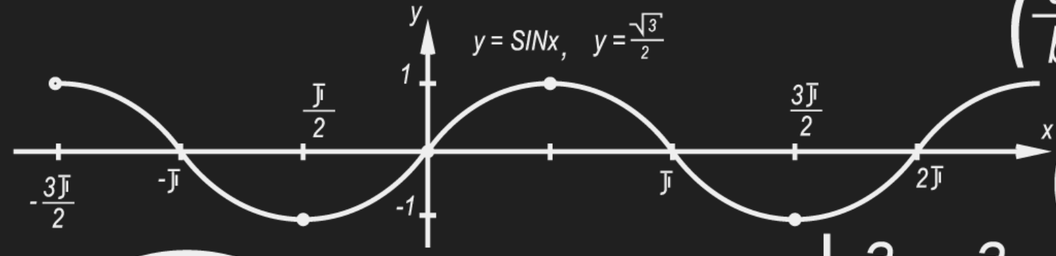
$$\frac{a}{SIN\ A} = \frac{b}{SIN\ B} = \frac{c}{SIN\ C} = 2R$$

$$A = \frac{1}{2}\ ap$$

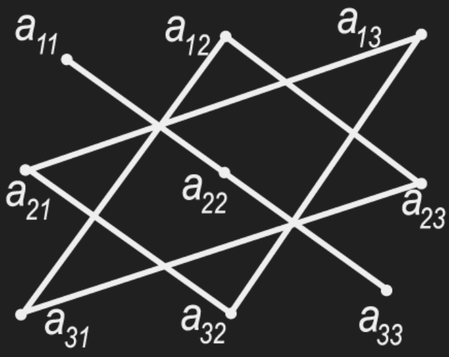


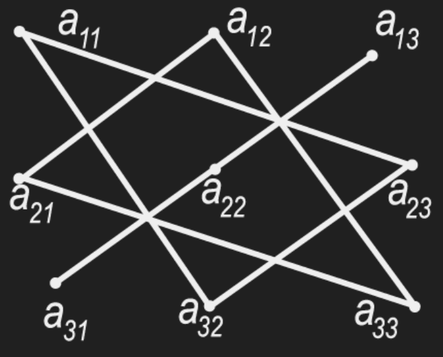
$$\sqrt{a}\ \sqrt{b} = \sqrt{a\,b}$$

$$\left(\frac{a}{b}\right)^{-n} = \left(\frac{b}{a}\right)^{n}$$



$$\left(\frac{a}{b}\right)^{n} = \frac{a^n}{b^n}$$





$$(\ln x)' = \frac{1}{x}$$

$$(kx+b) = k.$$

$$(x')' = rx^{r-1}$$

$$(e^x)' = e^x$$

$$(C)' = 0.$$

$$\Delta_3 = |A| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\,a_{22}\,a_{33} + a_{12}\,a_{23}\,a_{31} + a_{13}\,a_{21}\,a_{32} - a_{13}\,a_{22}\,a_{31} - a_{12}\,a_{21}\,a_{33} - a_{11}\,a_{23}\,a_{32}$$

$$V = \frac{4}{3}\ \pi r^3$$

$$V = \frac{1}{3}\ \pi r^2 h$$

$$c^2 = a^2 + b^2 - 2ab\ COS\ C$$

$$\log_a (x^p) = p \log_a (x)$$

$$z = re^{i\varphi}$$

$$\log_a |xy| = \log_a |x| + \log_a |y|$$

$$a^{\frac{1}{n}} = \sqrt[n]{a}$$

$$\varphi(x) = \lambda \int_a^b K(x, s)\, \varphi(s)\, ds + f(x)$$

$$\frac{d}{dx} \ln x = \frac{1}{x}$$

$$\int \ln x\, dx = x \ln x - x + C$$

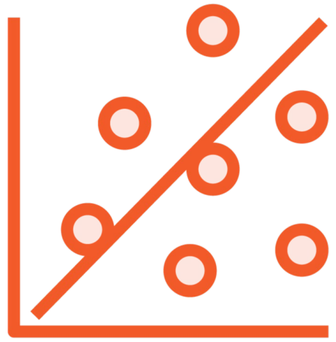
$$\varphi(x) = \int K(x, s, \varphi(s))\, ds,$$

$$\frac{dx}{dt} = F(t, x(t)), \quad x(a) = x_0$$

Regression is the process for estimating the relationship between a continuous response variable ***y*** and a set of variables ***X*** that describe ***y***

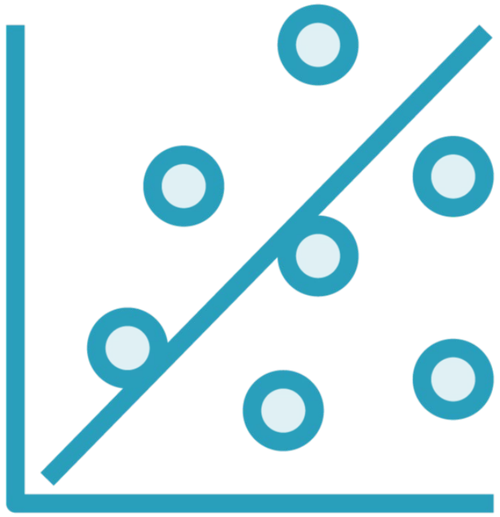
Linear Regression

Linear Regression



The process where we estimate a *linear* relationship between one or more explanatory variables (*X*) and a scalar response (*y*)

Linear Regression



- **Predicts continuous values**
- **Assumes a linear relationship between *X* and *y***
- **Sensitive to outliers**
- **Expects numerical features**
- **Parametric algorithm**

$$\hat{y} = \beta_0 * 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

- Formula used for prediction

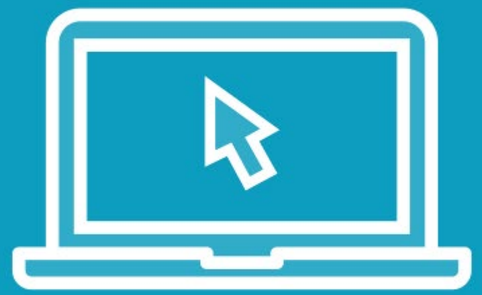
$$\hat{\beta} = (X^T X)' X^T y$$

- Obtain Beta coefficients

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Use least squares to find the difference between original and predicted

Demo

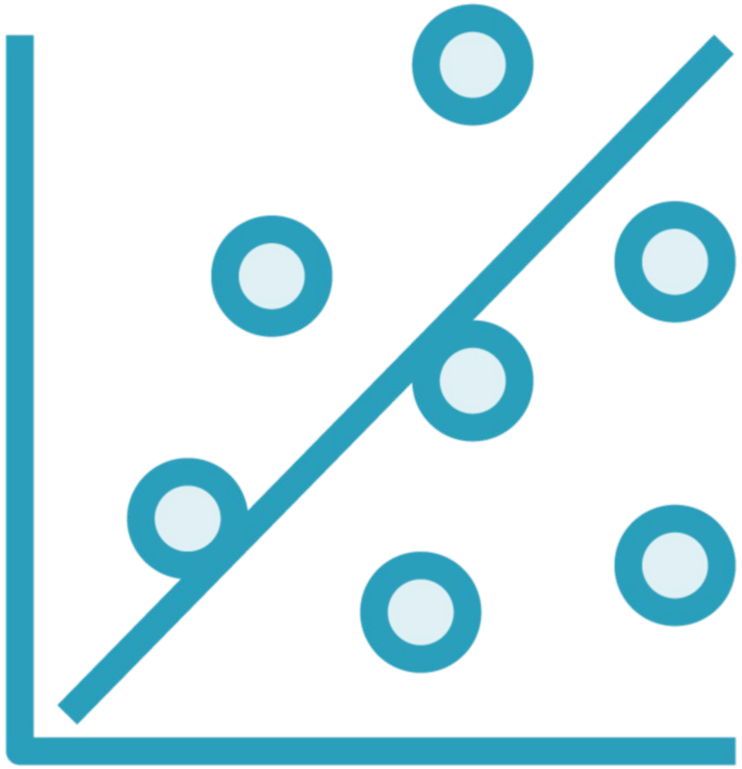


Linear Regression

Polynomial Regression

Explaining Data

Using a line



Using a curve

Polynomial Regression

Is an extended linear regression case

Where the relationship is an n^{th} degree polynomial

A curve function shows a high-level correlation

Plotting helps identify the linear or curve relationship

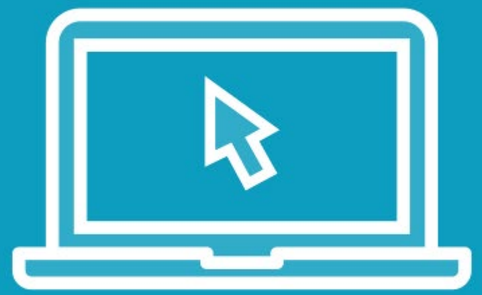
$$\hat{y} = \beta_0 * 1 + \beta_1 x_1 + \beta_2 x_2^2 + \ldots + \beta_k x_k^n$$

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

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Formula used for prediction with n degree polynomial
- Obtain Beta coefficients
- Use least squares to find the difference between original and predicted

Demo

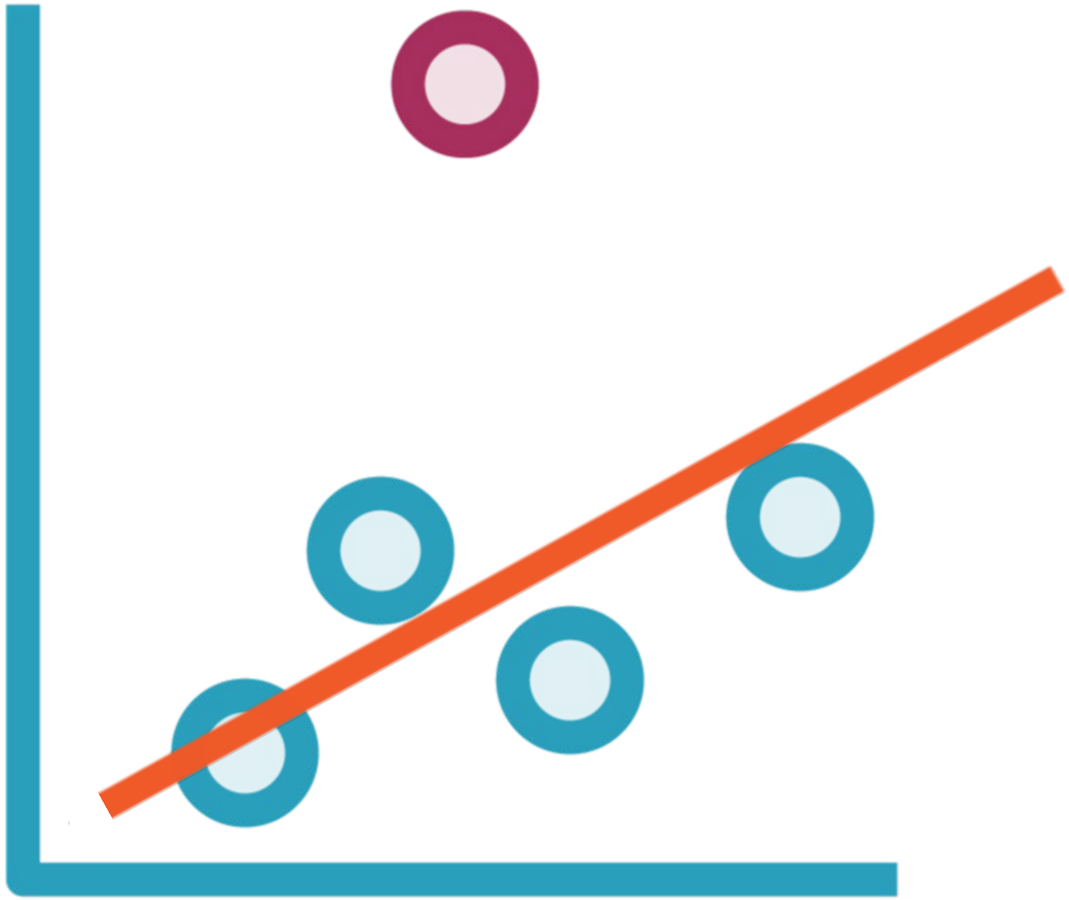


Polynomial Regression

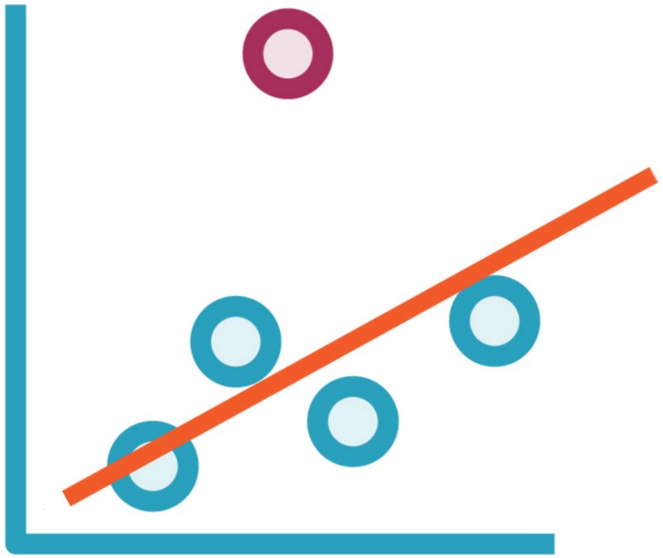
Lasso Regression

Linear Regression

Lasso Regression



Lasso (L1) Regression



Reduces number of features from your dataset

- Some of them may turn to zero

Tries to avoid overfitting and improves generalization

Works both with linear and polynomial regression

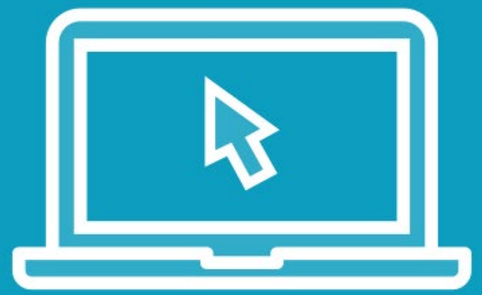
$$\hat{y} = \beta_0 * 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

$$\hat{y}_i = \beta_0 + \sum_{i=1}^{m} x_i \beta_i$$

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |\beta_j|$$

- Remember the original linear regression formula?
- Rewrite it like this
- Reuse the cost function and add L1 penalty term

Demo



Lasso (L1) Regression

Ridge Regression

Lasso (L1) vs. Ridge (L2) Regression

Lasso (L1)

Leads to some coefficients to be zero

Removing them completely

Works well with a small number of parameters that are significant

Many other parameters do not influence

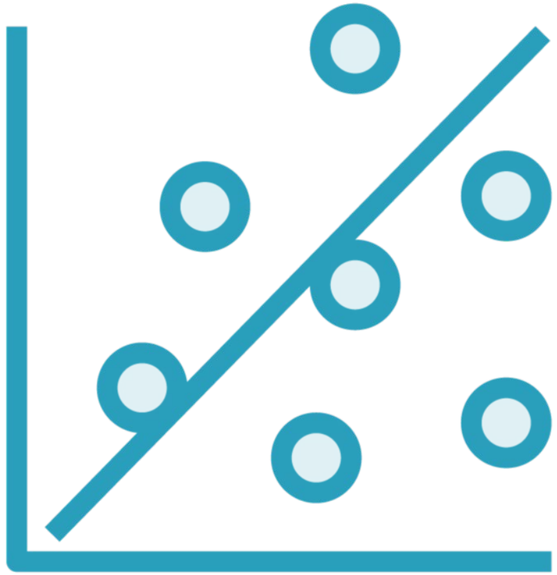
Ridge (L2)

Coefficients can take near-to-zero values

But not removed altogether

Leaves parameters with minimal values

Ridge (L2) Regression



Keeps all features in the model

Does not perform feature selection like Lasso

Helpful when you need to use all features in the model

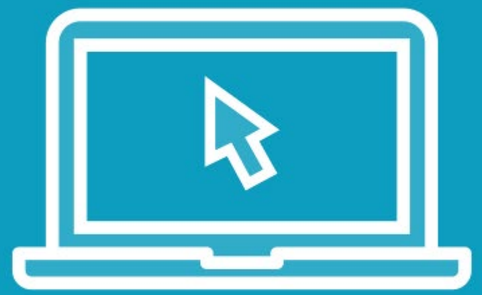
$$\hat{y} = \beta_0 * 1 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

$$\hat{y}_i = \beta_0 + \sum_{i=1}^{m} x_i \beta_i$$

$$J(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} (\beta_j)^2$$

- Remember the original linear regression formula?
- Rewrite it like this
- Reuse the cost function and add L2 penalty term

Demo



Ridge (L2) Regression

Perceptron Regression

Perceptron Regression

Perceptron Regression

Requires matrix multiplications

- Instead of arrays (linear regression)

Can be transformed into a classifier

Weights usually obtained with an optimization algorithm

- In contrast, linear regression uses least squares

In the output node, perceptron works exactly like linear regression

$$h_{in}(X) = b_j + \sum_{i=1}^{n} w_{ji} x_i$$

- Hidden layer input calculations

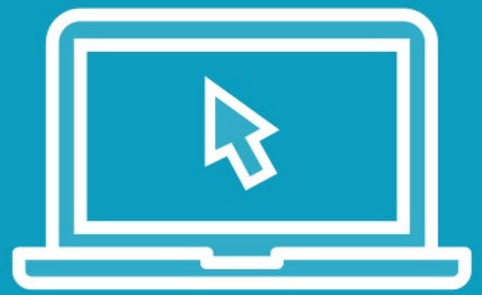
$$h_{out}(X) = \frac{1}{1 + e^{-h_{in}(X)}}$$

- Hidden layer **output** calculations (using sigmoid)

$$\hat{y}(X) = b + \sum_{j=1}^{m} w_j h_{out}(X)$$

- The final node prediction is similar to linear regression

Demo



Perceptron Regression

Takeaway

Regression algorithms predict continuous values

Prone to overfitting

Regularization is a way to deal with overfitting

- Use L1 or L2 if simple linear regression does not work very well

Remove outliers

Plotting data helps understand data shape